OBJECT ORIENTED PROGRAMMING
ETCS 258

Faculty name – Ms. Karuna

Student name: Syeda Reeha Quasar

Roll No.: 14114802719

Semester: 4th

Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 11008

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

MAIT

उद्यमेन हि सिध्यन्ति
कार्याणि न मनोरथैः

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence.

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware – Software Symbiosis

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life – Long Learning

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification

The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

Entrepreneurship

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens.

उद्यमेन हि सिध्यन्ति
कार्याणि न मनोरथैः

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

To Produce “Critical thinkers of Innovative Technology”

MISSION

To provide an excellent learning environment across the computer science discipline to inculcate professional behavior, strong ethical values, innovative research capabilitiesand leadership abilities which enable them to become successful entrepreneurs in this globalized world.

1. To nurture an **excellent learning environment** that helps students to enhance their problem solving skills and to prepare students to be lifelong learners by offering a solid theoretical foundation with applied computing experiences and educating them about their **professional, and ethical responsibilities**.
2. To establish **Industry-Institute Interaction**, making students ready for the industrial environment and be successful in their professional lives.
3. To promote **research activities** in the emerging areas of technology convergence.
4. To build engineers who can look into technical aspects of an engineering solution thereby setting a ground for producing successful **entrepreneur.**

INDEX

	which takes two argument is used to initialized real and imag to two different values.				
13.	Write a program to enter any number and find its factorial using constructor.	15-04-2021			
14.	Write a program to generate a Fibonacci series using Copy Constructor.	15-04-2021			
15.	Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the bigger of two entered numbers.	22-04-2021			
16.	Create a base class basic_info with data members name, roll no, sex and two member functions getdata and display. Derive a class physical_fit from basic_info which has data members height and weight and member functions getdata and display. Display all the information using object of derived class.	29-04-2021			
17.	Create class first with data members book no, book name and member function getdata and putdata. Create a class second with data members author name, publisher and members getdata and showdata. Derive a class third from first and second with data member no of pages and year of publication. Display all these information using array of objects of third class.	29-04-2021			
18.	Write a program to read the class object of student info such as name, age, sex, height and weight from the keyboard and to store them on a specified file using read () and write () functions. Again, the same file is opened for reading and displaying the contents of the file on the screen.	29-04-2021			
19.	Create a class called LIST with two pure virtual function store () and retrieve (). To store a value call store and to retrieve call retrieve function. Derive two classes stack and queue from it and override store and retrieve.	20-05-2021			
20.	Create a base class called SHAPE. Use this class to store two double type values. Derive two specific classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata to initialize base class data members and another member function display to compute and display the area of figures. Make display a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes design a program that will accept driven of a TRIANGLE or	20-05-2021			

	RECTANGLE interactively and display the area.				
21.	Design three classes STUDENT, EXAM, and RESULT. The STUDENT class has data members such as roll no, name. Create a class EXAM by inheriting the STUDENT class. The EXAM class adds data members representing the marks scored in six subjects. Derive the RESULT from the EXAM class and has its own data members such as total marks. WAP to model this relationship.	20-05-2021			
22.	Implement a class string containing the following functions: a. Overload + operator to carry out the concatenation of strings. b. Overload = operator to carry out string copy. c. Overload <= operator to carry out the comparison of strings. d. Function to display the length of string. e. Function tolower() to convert upper case to lower case. f. Function toupper() to convert lower case letters to upper case.	27-05-2021			
23.	Write a program to overload new and delete operator.	27-05-2021			
24.	Write a program to overload unary increment (++) operator.	27-05-2021			
25.	Write a program to raise an exception if any attempt is made to refer to an element whose index is beyond the array size.	03-06-2021			
26.	Write a program to read two numbers and then divide first no by second no and raise an exception if second number is zero.	03-06-2021			
27.	Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space, line feed, new line and carriage return from a text file and store the contents of the file without the white spaces on another file.	03-06-2021			
28.	Write a program to define the function template for calculating the square of given numbers with different data types.	03-06-2021			
29.	Write a program to define the function template for swapping two items of various datatypes such as integers, float and characters.	10-06-2021			
30.	Write a program to illustrate how Template function can be overloaded.	10-06-2021			

31.	Write a program to illustrate how to define and declare a class template for reading two data items from the keyboard and to find their sum.	10-06-2021			
32.	Write a program to read a set of lines from the keyboard and to store it on a specified file.	10-06-2021			
33.	Write a program to read a text file and display its contents on the screen.	10-06-2021			
34.	Write a program to copy the contents of a file into another.	10-06-2021			

EXPERIMENT - 1

Object Oriented Programming Lab

Aim

Write a program to find whether a number is prime or not.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 1

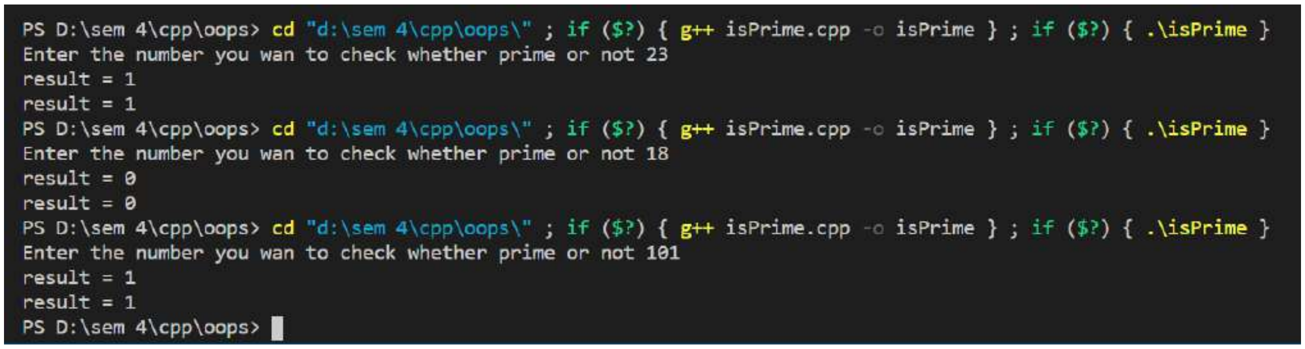
Aim:

Write a program to find whether a number is prime or not.

Source Code:

```
#include <iostream>
#include <math.h>
using namespace std;

bool isPrime(int n){
    if (n < 2) return false;
    for (int i = 2; i < n; ++i) {
        if ((n % i) == 0) return false;
    }
    return true;
}

bool isPrime1(int n){
    if (n < 2) return false;
    if (n % 2 == 0) return false;
    for (int i = 2; i < sqrt(n); ++i) {
        if ((n % i) == 0) return false;
    }
    return true;
}

int main() {
    cout << "Enter the number you wan to check whether prime or not";
    int n;
    cin >> n;
    cout << "result = " << isPrime(n);
    cout << "result = " << isPrime1(n);
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ isPrime.cpp -o isPrime } ; if ($?) { .\isPrime }
Enter the number you wan to check whether prime or not 23
result = 1
result = 1
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ isPrime.cpp -o isPrime } ; if ($?) { .\isPrime }
Enter the number you wan to check whether prime or not 18
result = 0
result = 0
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ isPrime.cpp -o isPrime } ; if ($?) { .\isPrime }
Enter the number you wan to check whether prime or not 101
result = 1
result = 1
PS D:\sem 4\cpp\oops>
```

Viva Questions

1. What are the different data types present in C++?

Ans.

The 4 data types in C++ are given below:

- Primitive Datatype(basic datatype). Example- char, short, int, float, long, double, bool, etc.
- Derived datatype. Example- array, pointer, etc.
- Enumeration. Example- enum
- User-defined data types. Example- structure, class, etc.

2. What is the difference between C and C++?

Ans.

The main difference between C and C++ are provided in the table below:

C	C++
C is a procedure-oriented programming language.	C++ is an object-oriented programming language.
C does not support data hiding.	Data is hidden by encapsulation to ensure that data structures and operators are used as intended.
C is a subset of C++	C++ is a superset of C.
Function and operator overloading are not supported in C	Function and operator overloading is supported in C++
Namespace features are not present in C	Namespace is used by C++, which avoids name collisions.
Functions can not be defined inside structures.	Functions can be defined inside structures.
calloc() and malloc() functions are used for memory allocation and free() function is used for memory deallocation.	new operator is used for memory allocation and deletes operator is used for memory deallocation.

3. What are class and object in C++?

Ans.

A class is a user-defined data type that has data members and member functions. Data members are the data variables and member functions are the functions that are used to perform operations on these variables.

An object is an instance of a class. Since a class is a user-defined data type so an object can also be called a variable of that data type.

EXPERIMENT - 2

Object Oriented Programming Lab

Aim

Write a program to take name, address as character string, age as int, salary as float and contains inline function to set the values and display them.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 2

Aim:

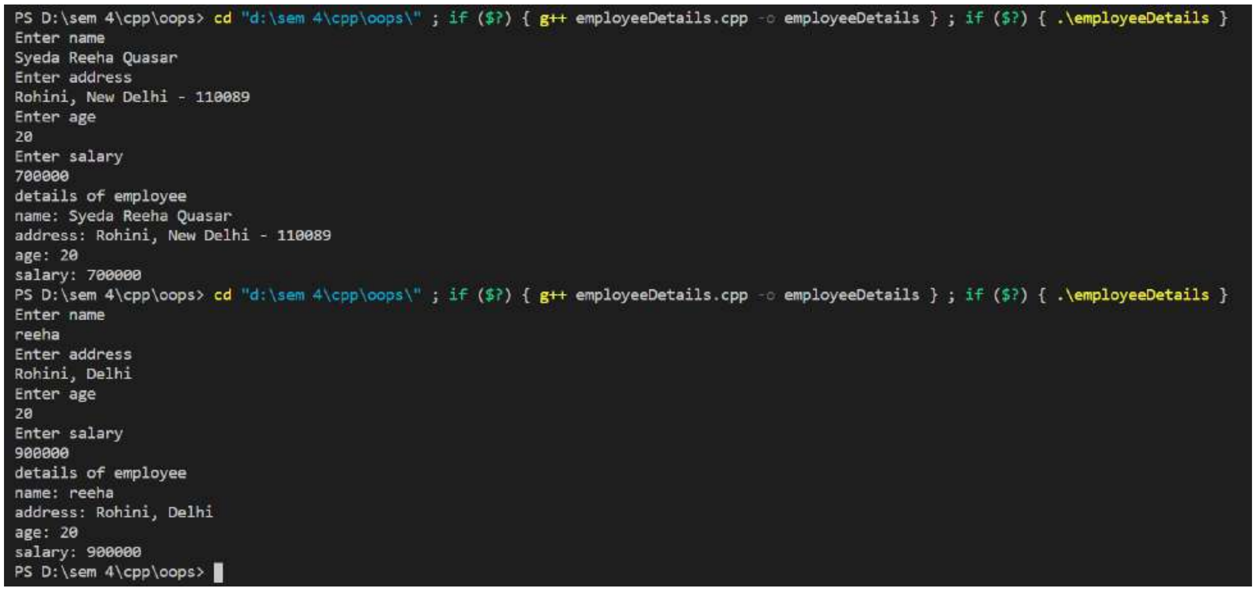
Write a program to take name, address as character string, age as int, salary as float and contains inline function to set the values and display them.

Source Code:

```
#include <iostream>
#include <string>
using namespace std;

class details{
    public:
        string name;
        string address;
        int age;
        float salary;

    void getData() {
        cout << "Enter name" << endl;
        getline(cin, name);
        cout << "Enter address" << endl;
        getline(cin, address);
        cout << "Enter age" << endl;
        cin >> age;
        cout << "Enter salary" << endl;
        cin >> salary;
    }

    void showdata();

};

void details :: showdata(){
    cout << "details of employee \n";
    cout << "name: " << name << endl;
    cout << "address: " << address << endl;
    cout << "age: " << age << endl;
    cout << "salary: " << salary << endl;
}
```

```
int main(){
    details d;
    d.getData();
    d.showdata();
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ employeeDetails.cpp -o employeeDetails } ; if ($?) { .\employeeDetails }
Enter name
Syeda Reeha Quasar
Enter address
Rohini, New Delhi - 110089
Enter age
20
Enter salary
700000
details of employee
name: Syeda Reeha Quasar
address: Rohini, New Delhi - 110089
age: 20
salary: 700000
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ employeeDetails.cpp -o employeeDetails } ; if ($?) { .\employeeDetails }
Enter name
reeha
Enter address
Rohini, Delhi
Enter age
20
Enter salary
900000
details of employee
name: reeha
address: Rohini, Delhi
age: 20
salary: 900000
PS D:\sem 4\cpp\oops>
```

Viva Questions

1. What are the C++ access specifiers?

Ans.

In C++ there are the following access specifiers:

Public: All data members and member functions are accessible outside the class.

Protected: All data members and member functions are accessible inside the class and to the derived class.

Private: All data members and member functions are not accessible outside the class.

2. Define inline function

Ans.

If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time. One of the important advantages of using an inline function is that it eliminates the function calling overhead of a traditional function.

3. What is a reference in C++?

Ans.

A reference is like a pointer. It is another name of an already existing variable. Once a reference name is initialized with a variable, that variable can be accessed by the variable name or reference name both.

EXPERIMENT - 3

Object Oriented Programming Lab

Aim

Using concept of function overloading, write function for calculating area of triangle, circle and rectangle.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 3

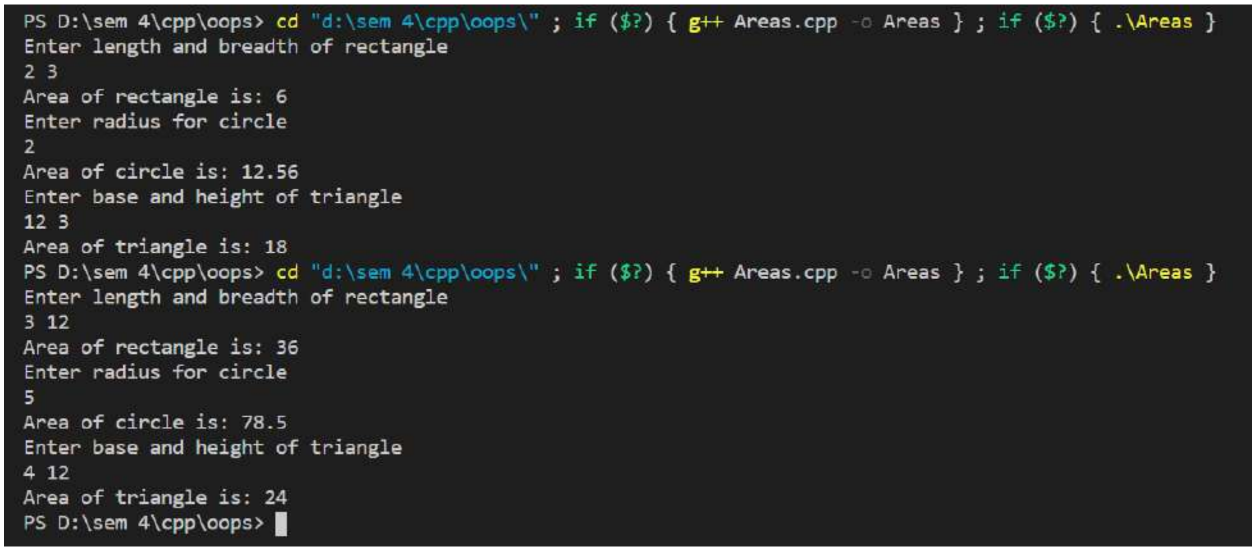
Aim:

Using concept of function overloading, write function for calculating area of triangle, circle and rectangle.

Source Code:

```
#include <iostream>
using namespace std;

int area(int l, int b){
    return l * b;
}

float area(float r){
    return 3.14 * r * r;
}

float area(float b, float h){
    return (b * h)/2;
}

int main(){
    int l, b;
    float r, ba, he;
    cout << "Enter length and breadth of rectangle" << endl;
    cin >> l >> b;
    cout << "Area of rectangle is: " << area(l, b) << endl;
    cout << "Enter radius for circle"<<endl;
    cin >> r;
    cout << "Area of circle is: " << area(r) << endl;
    cout << "Enter base and height of triangle" << endl;
    cin >> ba >> he;
    cout << "Area of triangle is: " << area(ba, he) << endl;
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ Areas.cpp -o Areas } ; if ($?) { .\Areas }
Enter length and breadth of rectangle
2 3
Area of rectangle is: 6
Enter radius for circle
2
Area of circle is: 12.56
Enter base and height of triangle
12 3
Area of triangle is: 18
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ Areas.cpp -o Areas } ; if ($?) { .\Areas }
Enter length and breadth of rectangle
3 12
Area of rectangle is: 36
Enter radius for circle
5
Area of circle is: 78.5
Enter base and height of triangle
4 12
Area of triangle is: 24
PS D:\sem 4\cpp\oops>
```

Viva Questions

1. What do you mean by abstraction in C++?

Ans.

Abstraction is the process of showing the essential details to the user and hiding the details which we don't want to show to the user or hiding the details which are irrelevant to a particular user.

2. Is deconstructor overloading possible? If yes then explain and if no then why?

Ans.

No destructor overloading is not possible. Destructors take no arguments, so there's only one way to destroy an object. That's the reason destructor overloading is not possible.

3. What do you mean by call by value and call by reference?

Ans.

In call by value method, we pass a copy of the parameter is passed to the functions. For these copied values a new memory is assigned and changes made to these values do not reflect the variable in the main function.

In call by reference method, we pass the address of the variable and the address is used to access the actual argument used in the function call. So changes made in the parameter alter the passing argument.

EXPERIMENT - 4

Object Oriented Programming Lab

Aim

Using concept of function overloading, write function for calculating area of triangle, circle and rectangle.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 4

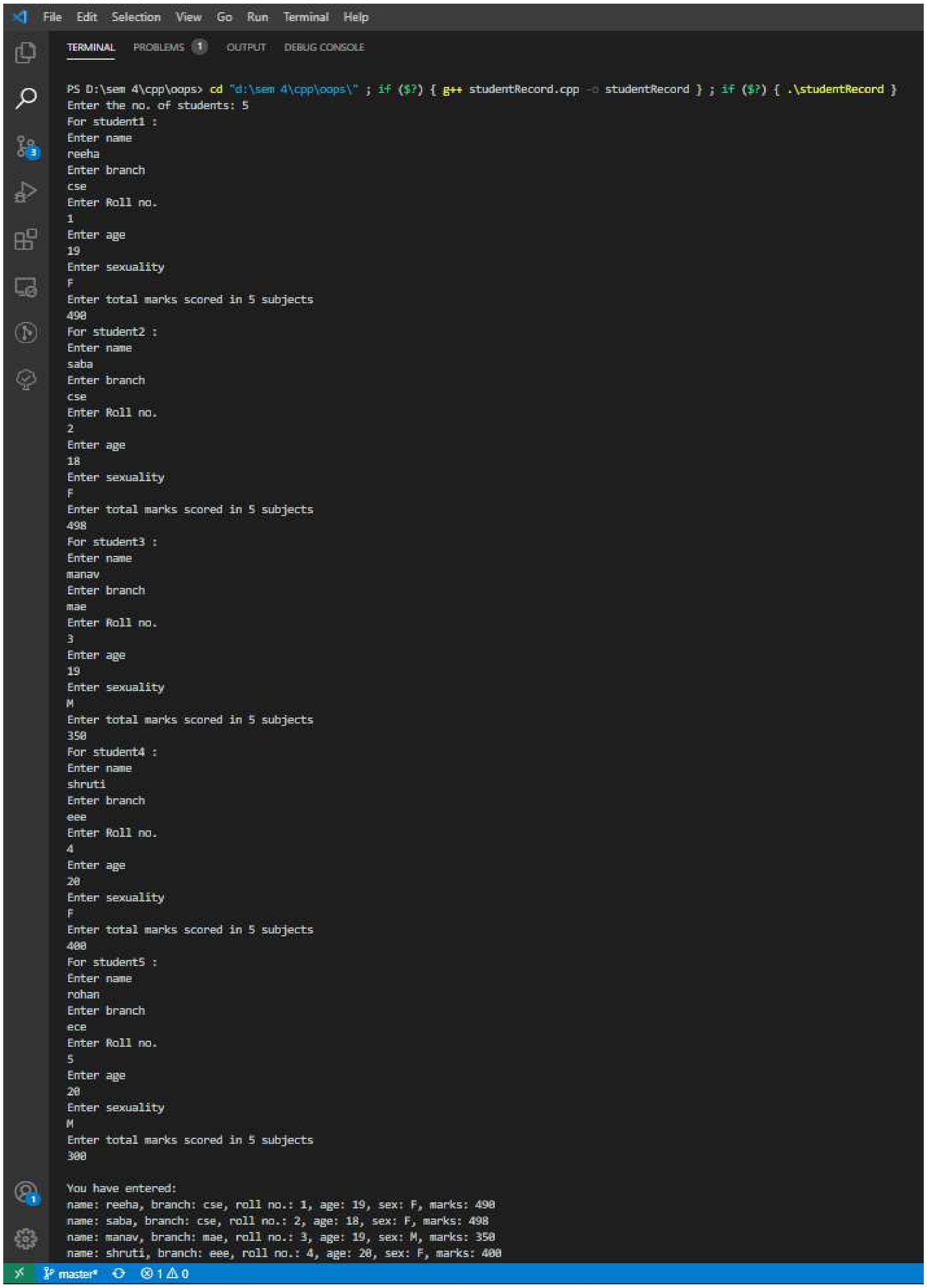
Aim:

Create a class student which have data members as name, branch, roll no., age, sex and marks in five subjects. Display the name of the student and his percentage who has more than 70%.

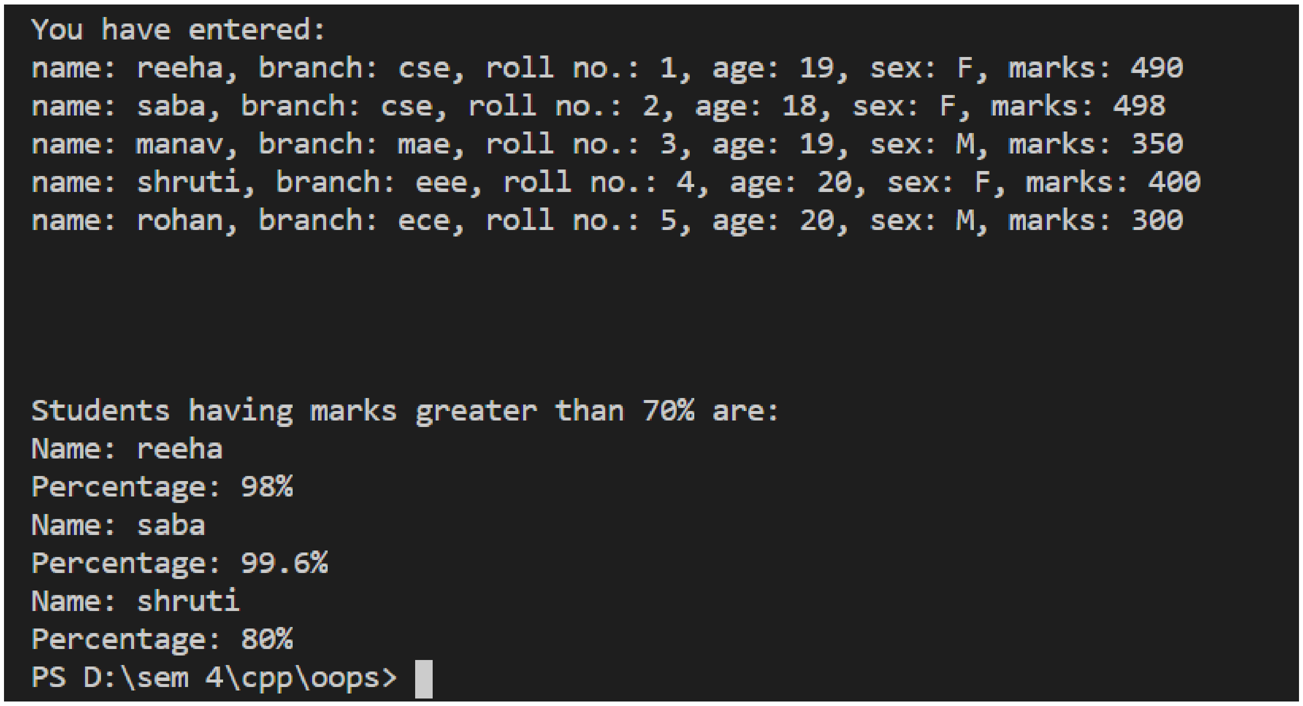
Source Code:

```
#include <iostream>
#include <string>

using namespace std;

class studentRecord{
    private:
        string name;
        string branch;
        int rollNo;
        int age;
        char sex[15];
        float marks;

    public:
        void getDetails();
        void check();
        void showDetails();
        void details();
};

void studentRecord::getDetails(){
    cout << "Enter name" << endl;
    cin >> name;
    cout << "Enter branch" << endl;
    cin >> branch;
    cout << "Enter Roll no. "<< endl;
    cin >> rollNo;
    cout << "Enter age" << endl;
    cin >> age;
    cout << "Enter sexuality" << endl;
    cin >> sex;
    cout << "Enter total marks scored in 5 subjects" << endl;
    cin >> marks;
}

void studentRecord::showDetails(){
    cout << "Name: " << name << endl;
    cout << "Percentage: " <<  (marks/5) << "%" << endl;
}

void studentRecord::check(){
    if (((marks/5)) > 70) {
        showDetails();
    }
}

void studentRecord::details(){
    cout << "name: " << name << ", branch: " << branch << ", roll no.: " << rollN
o << ", age: " << age << ", sex: " << sex << ", marks: " << marks << endl;
}

int main(int argc, char const *argv[]){
    int students;
    cout << "Enter the no. of students: ";
    cin >> students;
    studentRecord studentsArr[students];
    for (int i = 0; i < students; ++i) {
        cout << "For student" << i + 1 << " :" << endl;
        studentsArr[i].getDetails();
    }

    cout << "\nYou have entered:" << endl;
    for (int i = 0; i < students; i++) {
        studentsArr[i].details();
    }

    cout << "\n\n\n" << endl;

    cout << "Students having marks greater than 70% are: " << endl;
    for (int i = 0; i < students; i++) {
        studentsArr[i].check();
    }
}
```

Output:


File Edit Selection View Go Run Terminal Help
TERMINAL PROBLEMS 1 OUTPUT DEBUG CONSOLE
master* ⊗ 1 ⚠ 0

```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ studentRecord.cpp -o studentRecord } ; if ($?) { .\studentRecord }
Enter the no. of students: 5
For student1 :
Enter name
reeha
Enter branch
cse
Enter Roll no.
1
Enter age
19
Enter sexuality
F
Enter total marks scored in 5 subjects
490
For student2 :
Enter name
saba
Enter branch
cse
Enter Roll no.
2
Enter age
18
Enter sexuality
F
Enter total marks scored in 5 subjects
498
For student3 :
Enter name
manav
Enter branch
mae
Enter Roll no.
3
Enter age
19
Enter sexuality
M
Enter total marks scored in 5 subjects
350
For student4 :
Enter name
shruti
Enter branch
eee
Enter Roll no.
4
Enter age
20
Enter sexuality
F
Enter total marks scored in 5 subjects
400
For student5 :
Enter name
rohan
Enter branch
ece
Enter Roll no.
5
Enter age
20
Enter sexuality
M
Enter total marks scored in 5 subjects
300

You have entered:
name: reeha, branch: cse, roll no.: 1, age: 19, sex: F, marks: 490
name: saba, branch: cse, roll no.: 2, age: 18, sex: F, marks: 498
name: manav, branch: mae, roll no.: 3, age: 19, sex: M, marks: 350
name: shruti, branch: eee, roll no.: 4, age: 20, sex: F, marks: 400
```



```
You have entered:
name: reeha, branch: cse, roll no.: 1, age: 19, sex: F, marks: 490
name: saba, branch: cse, roll no.: 2, age: 18, sex: F, marks: 498
name: manav, branch: mae, roll no.: 3, age: 19, sex: M, marks: 350
name: shruti, branch: eee, roll no.: 4, age: 20, sex: F, marks: 400
name: rohan, branch: ece, roll no.: 5, age: 20, sex: M, marks: 300




Students having marks greater than 70% are:
Name: reeha
Percentage: 98%
Name: saba
Percentage: 99.6%
Name: shruti
Percentage: 80%
PS D:\sem 4\cpp\oops>
```

Viva Questions

1. What is an abstract class and when do you use it?

Ans.

A class is called an abstract class whose objects can never be created. Such a class exists as a parent for the derived classes. We can make a class abstract by placing a pure virtual function in the class.

2. What are destructors in C++?

Ans.

A constructor is automatically called when an object is first created. Similarly when an object is destroyed a function called destructor automatically gets called. A destructor has the same name as the constructor (which is the same as the class name) but is preceded by a tilde.

EXPERIMENT - 5

Object Oriented Programming Lab

Aim

Write a program for multiplication of two matrices using OOP.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 5

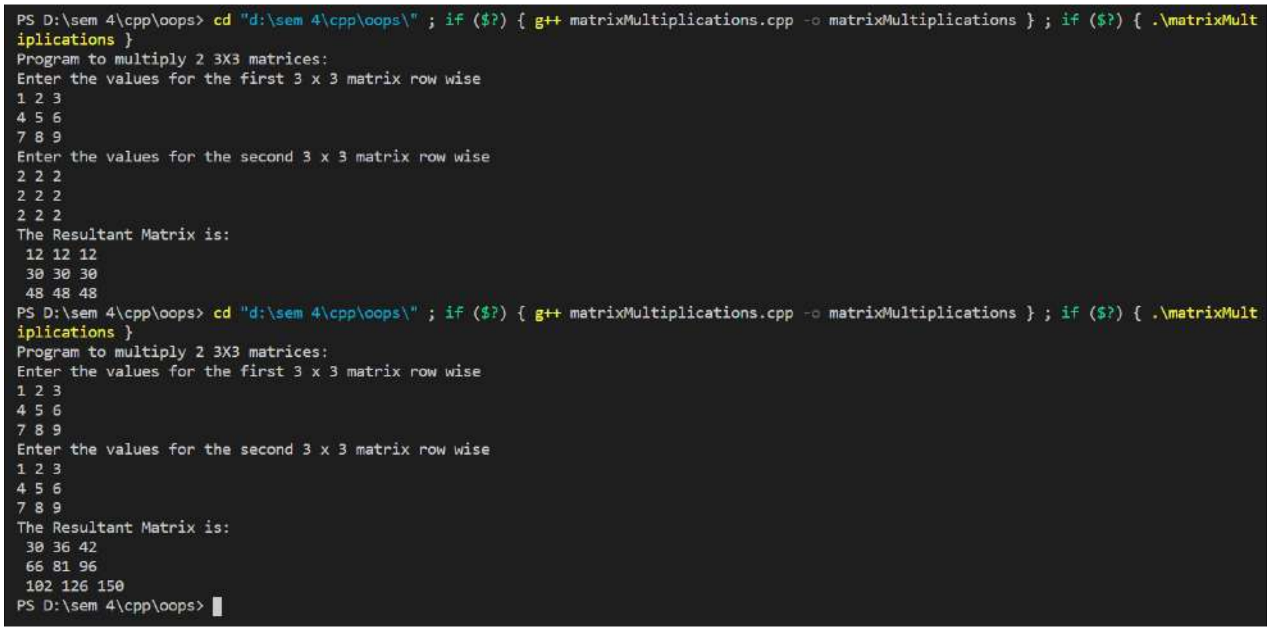
Aim:

Write a program for multiplication of two matrices using OOP.

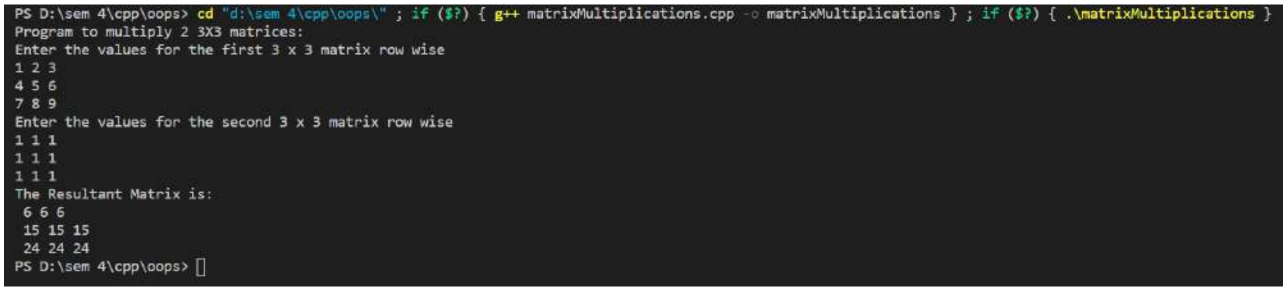
Source Code:

```
#include <iostream>

using namespace std;

class MatrixMultiplication{
    public:
        int a[3][3];
        int b[3][3];
        int c[3][3];

        void InputMatrix();
        void multiply();
        void result();
};

void MatrixMultiplication::InputMatrix(){
    cout << "Enter the values for the first 3 x 3 matrix row wise" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++)
        {
            cin >> a[i][j];
        }
    }
    cout << "Enter the values for the second 3 x 3 matrix row wise" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> b[i][j];
        }
    }
}

void MatrixMultiplication::multiply(){
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            c[i][j]=0;
            for (int k = 0; k < 3; k++) {
                c[i][j] += a[i][k] * b[k][j];
            }
        }
    }
}

void MatrixMultiplication::result(){
    cout << "The Resultant Matrix is: \n";
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << " " << c[i][j];
        }
        cout << endl;
    }
}

int main(){
    MatrixMultiplication x;
    cout << "Program to multiply 2 3X3 matrices: " << endl;
    x.InputMatrix();
    x.multiply();
    x.result();
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ matrixMultiplications.cpp -o matrixMultiplications } ; if ($?) { .\matrixMult
iplications }
Program to multiply 2 3X3 matrices:
Enter the values for the first 3 x 3 matrix row wise
1 2 3
4 5 6
7 8 9
Enter the values for the second 3 x 3 matrix row wise
2 2 2
2 2 2
2 2 2
The Resultant Matrix is:
 12 12 12
 30 30 30
 48 48 48
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ matrixMultiplications.cpp -o matrixMultiplications } ; if ($?) { .\matrixMult
iplications }
Program to multiply 2 3X3 matrices:
Enter the values for the first 3 x 3 matrix row wise
1 2 3
4 5 6
7 8 9
Enter the values for the second 3 x 3 matrix row wise
1 2 3
4 5 6
7 8 9
The Resultant Matrix is:
 30 36 42
 66 81 96
 102 126 150
PS D:\sem 4\cpp\oops>
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ matrixMultiplications.cpp -o matrixMultiplications } ; if ($?) { .\matrixMultiplications }
Program to multiply 2 3X3 matrices:
Enter the values for the first 3 x 3 matrix row wise
1 2 3
4 5 6
7 8 9
Enter the values for the second 3 x 3 matrix row wise
1 1 1
1 1 1
1 1 1
The Resultant Matrix is:
 6 6 6
 15 15 15
 24 24 24
PS D:\sem 4\cpp\oops>
```

Viva Questions

1) What is a class?

Ans.

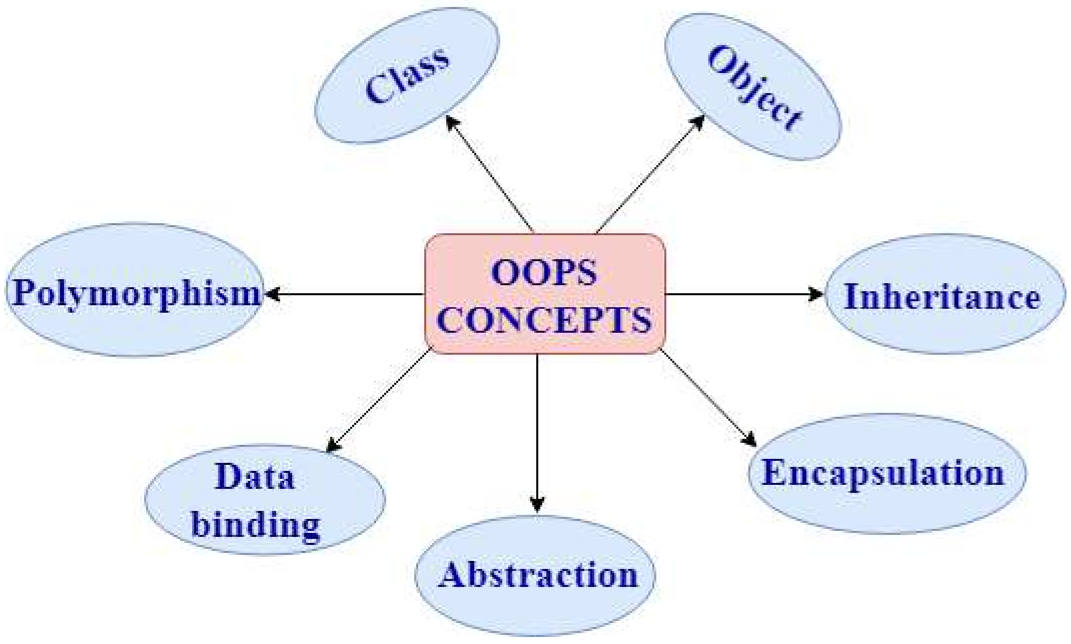
The class is a user-defined data type. The class is declared with the keyword class. The class contains the data members, and member functions whose access is defined by the three modifiers are private, public and protected. The class defines the type definition of the category of things. It defines a datatype, but it does not define the data it just specifies the structure of data.

You can create N number of objects from a class.

2) What are the various OOPs concepts in C++?

Ans.

The various OOPS concepts in C++ are:



- **Class:**

The class is a user-defined data type which defines its properties and its functions. For example, Human being is a class. The body parts of a human being are its properties, and the actions performed by the body parts are known as functions. The class does not occupy any memory space. Therefore, we can say that the class is the only logical representation of the data.

The syntax of declaring the class:

```
class student
{
//data members;
//Member functions
}
```

- **Object:**

An object is a run-time entity. An object is the instance of the class. An object can represent a person, place or any other item. An object can operate on both data members and member functions. The class does not occupy any memory space. When an object is created using a new keyword, then space is allocated for the variable in a heap, and the starting address is stored in the stack memory. When an object is created without a new keyword, then space is not allocated in the heap memory, and the object contains the null value in the stack.

```
class Student
{
//data members;
//Member functions
}
```

The syntax for declaring the object:

```
Student s = new Student();
```

- **Inheritance:**

Inheritance provides reusability. Reusability means that one can use the functionalities of the existing class. It eliminates the redundancy of code. Inheritance is a technique of deriving a new class from the old class. The old class is known as the base class, and the new class is known as derived class.

Syntax

```
class derived_class :: visibility-mode base_class;
```

Note: The visibility-mode can be public, private, protected.

- **Encapsulation:**

Encapsulation is a technique of wrapping the data members and member functions in a single unit. It binds the data within a class, and no outside method can access the data. If the data member is private, then the member function can only access the data.

- **Abstraction:**

Abstraction is a technique of showing only essential details without representing the implementation details. If the members are defined with a public keyword, then the members are accessible outside also. If the members are defined with a private keyword, then the members are not accessible by the outside methods.

- **Data binding:**

Data binding is a process of binding the application UI and business logic. Any change made in the business logic will reflect directly to the application UI.

- **Polymorphism:**

Polymorphism means multiple forms. Polymorphism means having more than one function with the same name but with different functionalities. Polymorphism is of two types:

1. Static polymorphism is also known as early binding.
2. Dynamic polymorphism is also known as late binding.

EXPERIMENT - 6

Object Oriented Programming Lab

Aim

To create a class TIME with members hours, minutes and seconds. Take input, add two-time objects and passing objects to function and display the result.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 6

Aim:

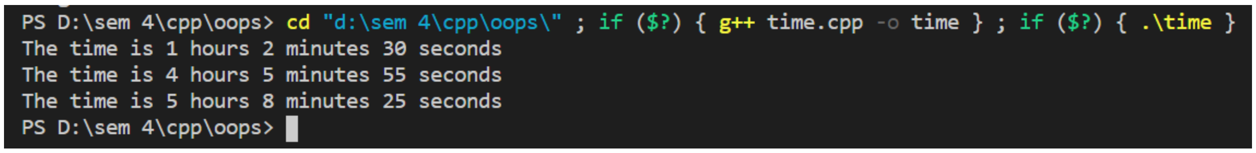
To create a class TIME with members hours, minutes and seconds. Take input, add two-time objects and passing objects to function and display the result.

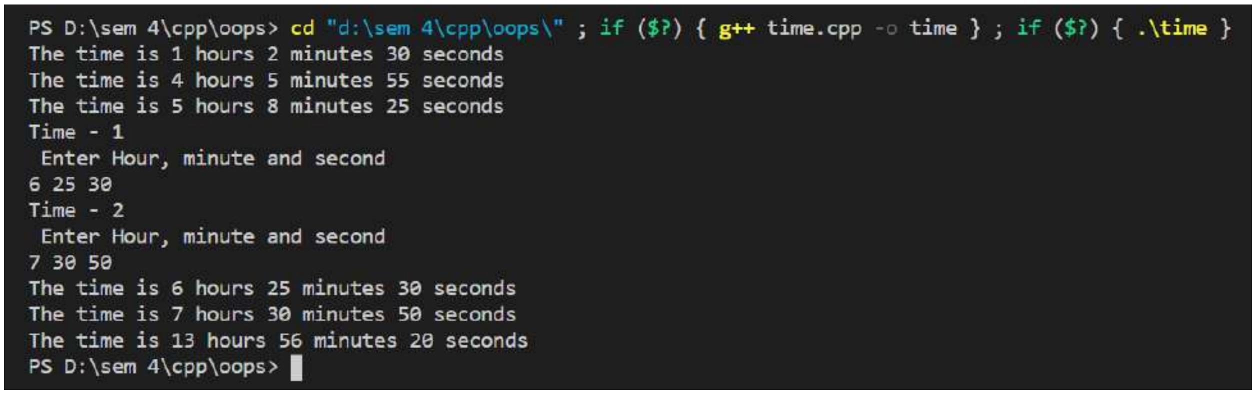
Source Code:

```
#include <iostream>

using namespace std;

class times{

    int seconds, minutes, hours;


    public:

        void getTime(int h, int m, int s){

            hours = h;

            minutes = m;

            seconds = s;

        }


        void putTime(){

            cout << "The time is " << hours << " hours " << minutes << " minutes
" << seconds << " seconds\n";

        }


        void sum(times, times);

};


void times::sum(times t1, times t2){

    seconds = t1.seconds + t2.seconds;

    minutes = seconds / 60;
```

```cpp
    seconds %= 60;
    minutes += t1.minutes + t2.minutes;
    hours = minutes / 60;
    minutes %= 60;
    hours += t1.hours + t2.hours;
}

int main(){
    times t1, t2, t3;
    t1.getTime(1, 2, 30);
    t2.getTime(4, 5, 55);
    t3.sum(t1, t2);
    t1.putTime();
    t2.putTime();
    t3.putTime();
    int hour1, minute1, second1, hour2, minute2, second2;
    cout << "Time - 1 \n Enter Hour, minute and second " << endl;
    cin >> hour1 >> minute1 >> second1;
    cout << "Time - 2 \n Enter Hour, minute and second " << endl;
    cin >> hour2 >> minute2 >> second2;
    t1.getTime(hour1, minute1, second1);
    t2.getTime(hour2, minute2, second2);
    t3.sum(t1, t2);
    t1.putTime();
    t2.putTime();
    t3.putTime();
    return 0;
}
```

Output:


```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ time.cpp -o time } ; if ($?) { .\time }
The time is 1 hours 2 minutes 30 seconds
The time is 4 hours 5 minutes 55 seconds
The time is 5 hours 8 minutes 25 seconds
PS D:\sem 4\cpp\oops> 
```


```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ time.cpp -o time } ; if ($?) { .\time }
The time is 1 hours 2 minutes 30 seconds
The time is 4 hours 5 minutes 55 seconds
The time is 5 hours 8 minutes 25 seconds
Time - 1
 Enter Hour, minute and second
6 25 30
Time - 2
 Enter Hour, minute and second
7 30 50
The time is 6 hours 25 minutes 30 seconds
The time is 7 hours 30 minutes 50 seconds
The time is 13 hours 56 minutes 20 seconds
PS D:\sem 4\cpp\oops> 
```

Viva Questions

1. What do you mean by abstraction in C++?

Ans.

Abstraction is the process of showing the essential details to the user and hiding the details which we don't want to show to the user or hiding the details which are irrelevant to a particular user.

2. Is deconstructor overloading possible? If yes then explain and if no then why?

Ans.

No destructor overloading is not possible. Destructors take no arguments, so there's only one way to destroy an object. That's the reason destructor overloading is not possible.

3. What do you mean by call by value and call by reference?

Ans.

In call by value method, we pass a copy of the parameter is passed to the functions. For these copied values a new memory is assigned and changes made to these values do not reflect the variable in the main function.

In call by reference method, we pass the address of the variable and the address is used to access the actual argument used in the function call. So changes made in the parameter alter the passing argument.

EXPERIMENT - 7

Object Oriented Programming Lab

Aim

To create a function power to raise a number m to a power n. The function takes double value for m and integer value for n. Use default value for n to make the function. Calculate the squares when this argument is omitted.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 7

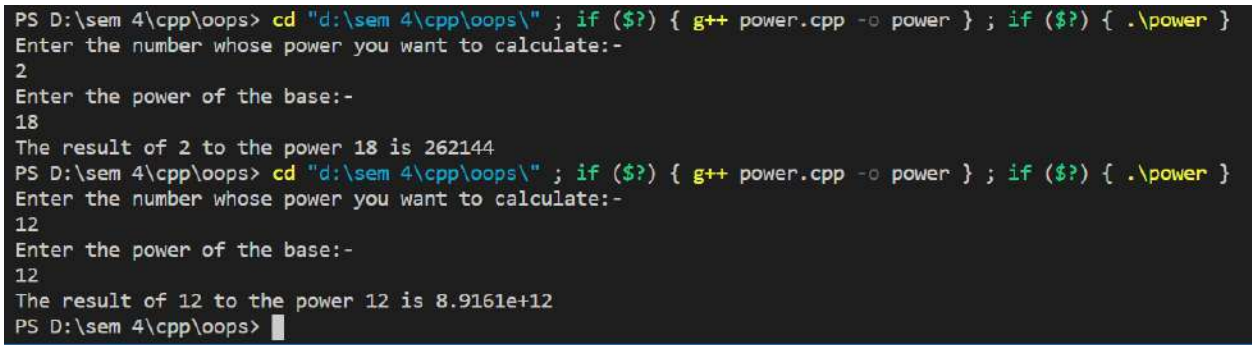
Aim:

To create a function power to raise a number m to a power n. The function takes double value for m and integer value for n. Use default value for n to make the function. Calculate the squares when this argument is omitted.

Source Code:

```cpp
#include <iostream>
using namespace std;

float power(double base, int powr = 0){
    if (powr == 0) {
        return base * base;
    }
    float res = 1;
    for (int i = 0; i < powr; ++i) {
        res *= base;
    }
    return res;
}

int main(){
    int powr;
    double base;
    cout << "Enter the number whose power you want to calculate:- " << endl;
    cin >> base;
    cout << "Enter the power of the base:- " << endl;
    cin >> powr;
    cout << "The result of " << base << " to the power " << powr << " is " << pow
er(base, powr) << endl;

    return 0;

}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ power.cpp -o power } ; if ($?) { .\power }
Enter the number whose power you want to calculate:-
2
Enter the power of the base:-
18
The result of 2 to the power 18 is 262144
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ power.cpp -o power } ; if ($?) { .\power }
Enter the number whose power you want to calculate:-
12
Enter the power of the base:-
12
The result of 12 to the power 12 is 8.9161e+12
PS D:\sem 4\cpp\oops>
```

Viva Questions

1. What is an abstract class and when do you use it?

Ans.

A class is called an abstract class whose objects can never be created. Such a class exists as a parent for the derived classes. We can make a class abstract by placing a pure virtual function in the class.

2. What are destructors in C++?

Ans.

A constructor is automatically called when an object is first created. Similarly when an object is destroyed a function called destructor automatically gets called. A destructor has the same name as the constructor (which is the same as the class name) but is preceded by a tilde.

EXPERIMENT - 8

Object Oriented Programming Lab

Aim

Write a program to find the greatest of two given numbers in two different classes using friend function.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 8

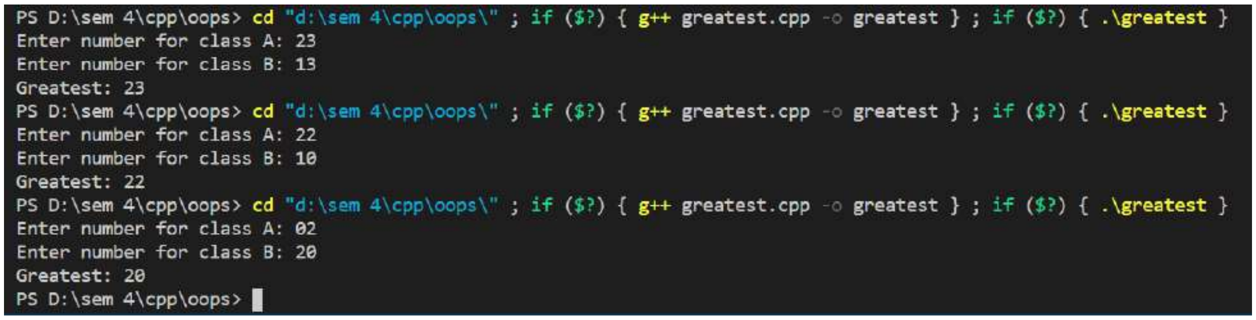
Aim:

Write a program to find the greatest of two given numbers in two different classes using friend function.

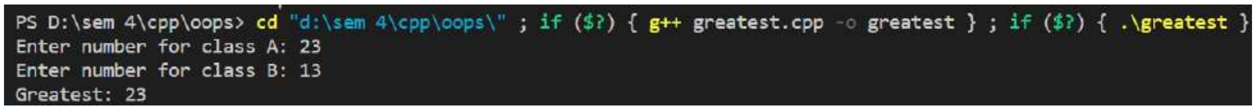
Source Code:

```
#include <iostream>

using namespace std;


class ClassB;


class ClassA {

    private:

        int numA;

        friend int greatest(ClassA, ClassB);


    public:

        void input() {

            cout << "Enter number for class A:";

            cin >> numA;

        }

};


class ClassB {

    private:

        int numB;

        friend int greatest(ClassA, ClassB);


    public:
        void input() {
            cout << "Enter number for class B:";
            cin >> numB;
        }
};

int greatest(ClassA objectA, ClassB objectB) {
    return max(objectA.numA, objectB.numB);
}

int main() {
    ClassA objectA;
    ClassB objectB;
    objectA.input();
    objectB.input();
    cout << "Greatest: " << greatest(objectA, objectB);
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ greatest.cpp -o greatest } ; if ($?) { .\greatest }
Enter number for class A: 23
Enter number for class B: 13
Greatest: 23
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ greatest.cpp -o greatest } ; if ($?) { .\greatest }
Enter number for class A: 22
Enter number for class B: 10
Greatest: 22
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ greatest.cpp -o greatest } ; if ($?) { .\greatest }
Enter number for class A: 02
Enter number for class B: 20
Greatest: 20
PS D:\sem 4\cpp\oops>
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ greatest.cpp -o greatest } ; if ($?) { .\greatest }
Enter number for class A: 23
Enter number for class B: 13
Greatest: 23
```

Viva Questions

Q1) What is an overflow error?

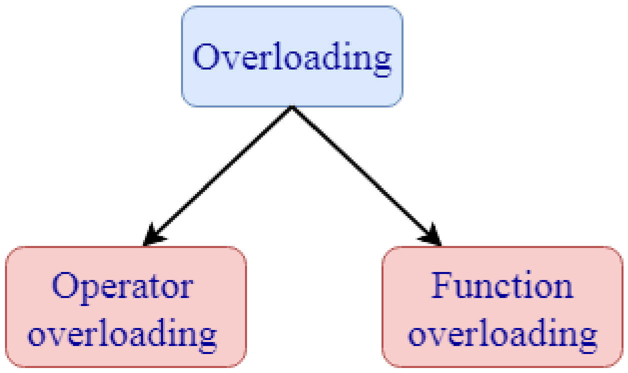
Ans.

It is a type of arithmetical error. It happens when the result of an arithmetical operation been greater than the actual space provided by the system.

Q2) What is overloading?

Ans.

- When a single object behaves in many ways is known as overloading. A single object has the same name, but it provides different versions of the same function.
- C++ facilitates you to specify more than one definition for a function name or an operator in the same scope. It is called function overloading and operator overloading respectively.
- **Overloading is of two types:**



1. **Operator overloading:** Operator overloading is a compile-time polymorphism in which a standard operator is overloaded to provide a user-defined definition to it. For example, '+' operator is overloaded to perform the addition operation on data types such as int, float, etc.

Operator overloading can be implemented in the following functions:

- Member function
- Non-member function
- Friend function

Syntax of Operator overloading:

```
Return_type classname :: Operator Operator_symbol(argument_list)
{
    // body_statements;
}
```

2. Function overloading: Function overloading is also a type of compile-time polymorphism which can define a family of functions with the same name. The function would perform different operations based on the argument list in the function call. The function to be invoked depends on the number of arguments and the type of the arguments in the argument list.

Q3) What is function overriding?

Ans.

If you inherit a class into a derived class and provide a definition for one of the base class's function again inside the derived class, then this function is called overridden function, and this mechanism is known as function overriding.

EXPERIMENT - 9

Object Oriented Programming Lab

Aim

Write a program to find the biggest of three number using Friend Function.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 9

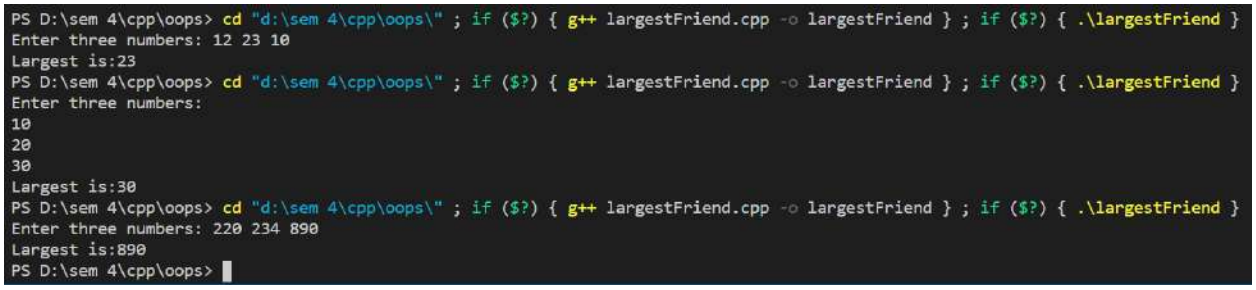
Aim:

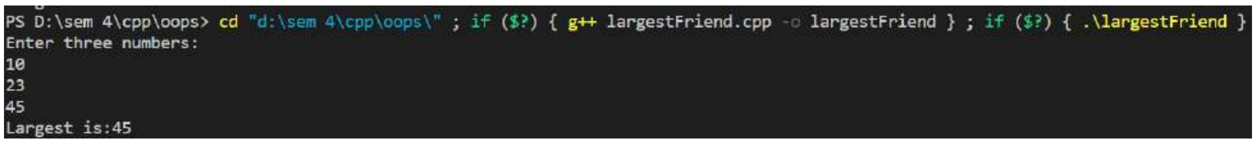
Write a program to find the biggest of three number using Friend Function.

Source Code:

```
#include <iostream>

using namespace std;


class threeNumbers{

    private:

        int x, y, z;


    public:

        void input(){

           cout << "Enter three numbers: ";

           cin >> x >> y >> z;

        }


        friend void findLargest(threeNumbers t);

};


void findLargest(threeNumbers t){

    if (t.x > t.y && t.x > t.z) {

        cout << "Largest is:" << t.x;

    }

    else if (t.y > t.z) {

        cout << "Largest is:" << t.y;

    }
    else {
        cout << "Largest is:" << t.z;
    }
}

int main(){
    threeNumbers t;
    t.input();
    findLargest(t);
    return 0;
}
```

Output:


```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ largestFriend.cpp -o largestFriend } ; if ($?) { .\largestFriend }
Enter three numbers: 12 23 10
Largest is:23
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ largestFriend.cpp -o largestFriend } ; if ($?) { .\largestFriend }
Enter three numbers:
10
20
30
Largest is:30
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ largestFriend.cpp -o largestFriend } ; if ($?) { .\largestFriend }
Enter three numbers: 220 234 890
Largest is:890
PS D:\sem 4\cpp\oops> 
```


```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ largestFriend.cpp -o largestFriend } ; if ($?) { .\largestFriend }
Enter three numbers:
10
23
45
Largest is:45
```

Viva Questions

Q1). What are friend functions?
Ans.

Friend Functions:

If a function is defined as a friend function then, the private and protected data of a class can be accessed using the function.

The complier knows a given function is a friend function by the use of the keyword friend.

For accessing the data, the declaration of a friend function should be made inside the body of the class (can be anywhere inside class either in private or public section) starting with keyword friend.

Q2). What are characteristics of friend function?
Ans.

Characteristics of friend function are as follows:

- friend function is not in the scope of the class in which it has been declared as friend.
- It cannot be called using the object of the class as it is not in the scope of the class.
- It can be called similar to a normal function without the help of any object.
- Unlike member functions, a friend function cannot access the member variables directly and has to use an object name and dot membership operator with each member variable.
- It can be declared either in the public or private scope area of a class.
- Usually, it has objects as arguments.

EXPERIMENT - 10

Object Oriented Programming Lab

Aim

Write a program to find the sum of two numbers declared in a class and display the numbers and sum using friend class.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 10

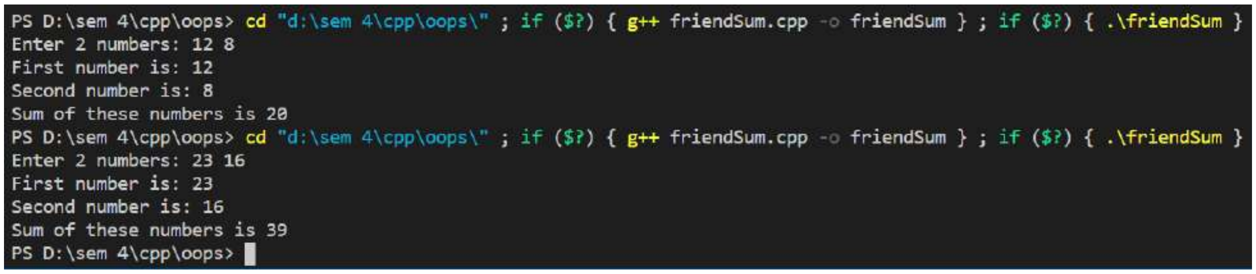
Aim:

Write a program to find the sum of two numbers declared in a class and display the numbers and sum using friend class.

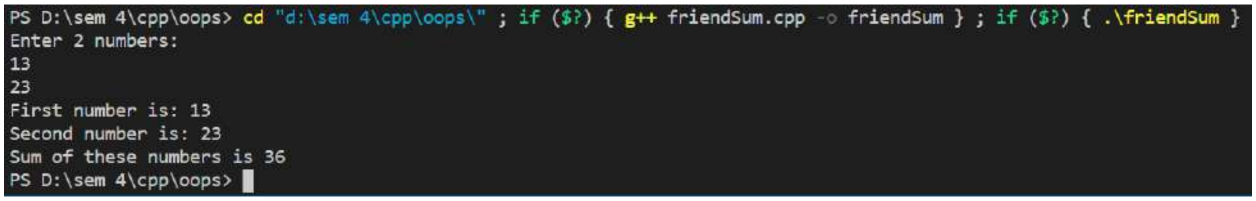
Source Code:

```
#include <iostream>

using namespace std;


class twoNum{

    private:

        int a, b;


    public:

        void input(){

           cout << "Enter 2 numbers: ";

           cin >> a >> b;

        }


        friend void sum(twoNum t);

};


void sum(twoNum t){

    cout << "First number is: " << t.a << endl;

    cout << "Second number is: " << t.b << endl;

    cout << "Sum of these numbers is " << t.a + t.b << endl;

}


int main(){
    twoNum t;

    t.input();

    sum(t);

    return 0;

}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ friendSum.cpp -o friendSum } ; if ($?) { .\friendSum }
Enter 2 numbers: 12 8
First number is: 12
Second number is: 8
Sum of these numbers is 20
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ friendSum.cpp -o friendSum } ; if ($?) { .\friendSum }
Enter 2 numbers: 23 16
First number is: 23
Second number is: 16
Sum of these numbers is 39
PS D:\sem 4\cpp\oops>
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ friendSum.cpp -o friendSum } ; if ($?) { .\friendSum }
Enter 2 numbers:
13
23
First number is: 13
Second number is: 23
Sum of these numbers is 36
PS D:\sem 4\cpp\oops>
```

Viva Questions

Q1. What are the benefits of friend function?

Ans.

- A friend function is used to access the non-public members of a class.
- It allows to generate more efficient code.
- It provides additional functionality which is not normally used by the class.
- It allows to share private class information by a non member function.
- It is used when two or more classes may contain members that are interrelated relative to others parts of the program.

Q2. What are disadvantages of friend function?

Ans.

- A derived class does not inherit friend function.
- Friend functions can not have a storage class specifier i.e they can not be declared as static or extern.

EXPERIMENT - 11

Object Oriented Programming Lab

Aim

Write a program to demonstrate the use of friend function with Inline assignment.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 11

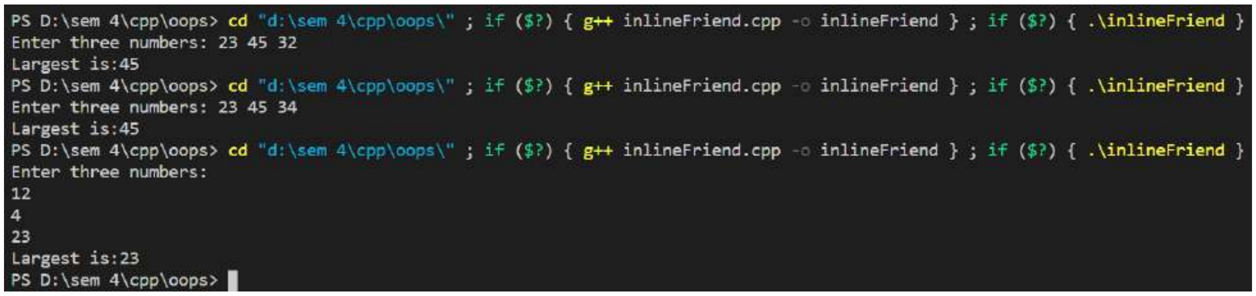
Aim:

Write a program to demonstrate the use of friend function with Inline assignment.

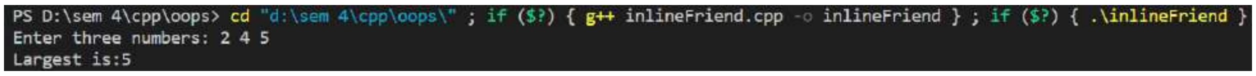
Source Code:

```
#include <iostream>

using namespace std;


class threeNumbers{

    private:

        int x, y, z;


    public:

        void input(){

           cout << "Enter three numbers: ";

           cin >> x >> y >> z;

        }


        friend inline void findLargest(threeNumbers t);

};


inline void findLargest(threeNumbers t){

    if (t.x > t.y && t.x > t.z) {

        cout << "Largest is:" << t.x;

    }

    else if (t.y > t.z) {

        cout << "Largest is:" << t.y;

    }
    else {
        cout << "Largest is:" << t.z;
    }
}

int main(){
    threeNumbers t;
    t.input();
    findLargest(t);
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ inlineFriend.cpp -o inlineFriend } ; if ($?) { .\inlineFriend }
Enter three numbers: 23 45 32
Largest is:45
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ inlineFriend.cpp -o inlineFriend } ; if ($?) { .\inlineFriend }
Enter three numbers: 23 45 34
Largest is:45
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ inlineFriend.cpp -o inlineFriend } ; if ($?) { .\inlineFriend }
Enter three numbers:
12
4
23
Largest is:23
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ inlineFriend.cpp -o inlineFriend } ; if ($?) { .\inlineFriend }
Enter three numbers: 2 4 5
Largest is:5
```

Viva Questions

Q1). What are friend functions?

Ans.

Friend Functions:

If a function is defined as a friend function then, the private and protected data of a class can be accessed using the function.

The complier knows a given function is a friend function by the use of the keyword friend.

For accessing the data, the declaration of a friend function should be made inside the body of the class (can be anywhere inside class either in private or public section) starting with keyword friend.

Q2). What are characteristics of friend function?

Ans.

Characteristics of friend function are as follows:

- friend function is not in the scope of the class in which it has been declared as friend.
- It cannot be called using the object of the class as it is not in the scope of the class.
- It can be called similar to a normal function without the help of any object.
- Unlike member functions, a friend function cannot access the member variables directly and has to use an object name and dot membership operator with each member variable.
- It can be declared either in the public or private scope area of a class.
- Usually, it has objects as arguments.

Q4. What are the benefits of friend function?

Ans.

- A friend function is used to access the non-public members of a class.

- It allows to generate more efficient code.
- It provides additional functionality which is not normally used by the class.
- It allows to share private class information by a non member function.
- It is used when two or more classes may contain members that are interrelated relative to others parts of the program.

Q5. What are disadvantages of friend function?

Ans.

- A derived class does not inherit friend function.
- Friend functions can not have a storage class specifier i.e they can not be declared as static or extern.

EXPERIMENT - 12

Object Oriented Programming Lab

Aim

Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and image parts to equal values and third which takes two argument is used to initialized real and image to two different values.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 12

Aim:

Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and image parts to equal values and third which takes two argument is used to initialized real and image to two different values.

Theory:

A class **constructor** is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Parameterized Constructor

A default constructor does not have any parameter, but if you need, a constructor can have parameters. This helps you to assign initial value to an object at the time of its creation.

Copy Constructor

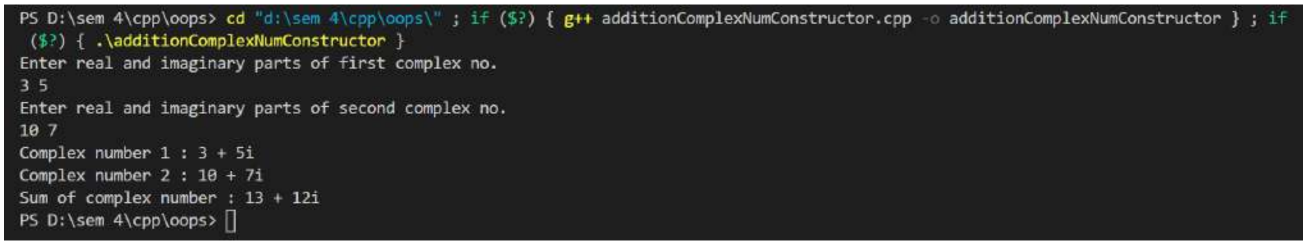
The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.

Source Code:

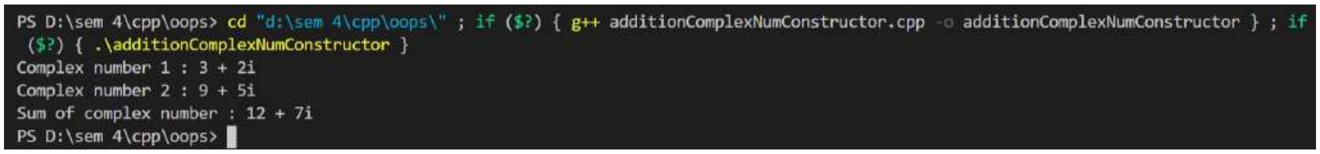
```
#include <iostream>
using namespace std;

class Complex{
    public:
        int real, imaginary; //declaration of variables to be used

        // Empty Constructor
        Complex(){}
        // Constructor to accept real and imaginary part
        Complex(int r, int i){
            real = r;
            imaginary = i;
        }

        // func for adding two complex number
        Complex addComp(Complex C1, Complex C2){
            Complex res;
            res.real = C1.real + C2.real;
            res.imaginary = C1.imaginary + C2.imaginary;
            return res;
        }
};

int main(){
    // // complex num 1
    // Complex C1(3, 2);
    // cout << "Complex number 1 : " << C1.real << " + " << C1.imaginary << "i" <
<endl;

    // // complex Num 2
    // Complex C2(9, 5);
    // cout << "Complex number 2 : " << C2.real << " + " << C2.imaginary << "i" <
< endl;

    // Complex C3;
    // C3 = C3.addComp(C1, C2);

    // cout << "Sum of complex number : " << C3.real << " + " << C3.imaginary <<
"i" << endl;

    int c1, r1, c2, r2;
    cout << "Enter real and imaginary parts of first complex no. " << endl;
    cin >> c1 >> r1;
    cout << "Enter real and imaginary parts of second complex no. " << endl;
    cin >> c2 >> r2;
    Complex C1(c1, r1);
    cout << "Complex number 1 : " << C1.real << " + " << C1.imaginary << "i" <<en
dl;

    // complex Num 2
    Complex C2(c2, r2);
    cout << "Complex number 2 : " << C2.real << " + " << C2.imaginary << "i" << e
ndl;
```

```
    Complex C3;
    C3 = C3.addComp(C1, C2);

    cout << "Sum of complex number : " << C3.real << " + " << C3.imaginary << "i"
 << endl;

}
```

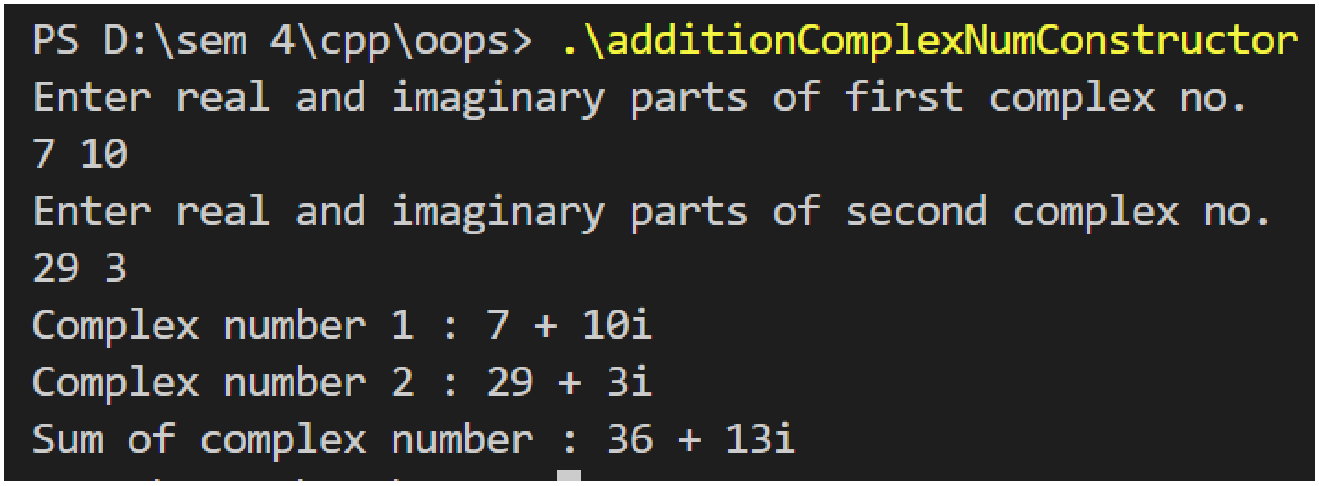
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ additionComplexNumConstructor.cpp -o additionComplexNumConstructor } ; if
 ($?) { .\additionComplexNumConstructor }
Enter real and imaginary parts of first complex no.
3 5
Enter real and imaginary parts of second complex no.
10 7
Complex number 1 : 3 + 5i
Complex number 2 : 10 + 7i
Sum of complex number : 13 + 12i
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ additionComplexNumConstructor.cpp -o additionComplexNumConstructor } ; if
 ($?) { .\additionComplexNumConstructor }
Complex number 1 : 3 + 2i
Complex number 2 : 9 + 5i
Sum of complex number : 12 + 7i
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> .\additionComplexNumConstructor
Enter real and imaginary parts of first complex no.
7 10
Enter real and imaginary parts of second complex no.
29 3
Complex number 1 : 7 + 10i
Complex number 2 : 29 + 3i
Sum of complex number : 36 + 13i
```

Viva Questions

Q1) What is a constructor?

Ans.

A **constructor** is a function of a class that has the same name as the class. The constructor is called at the time of the initialization of object. There are three types of constructors –

- Default constructor
- Parameterized constructor
- Copy constructor

Syntax

```
class cl_name{
        cl_name(){
        //This is constructor..
        }
}
```

Q2) What is a destructor?

Ans.

A **destructor** is a method of a class that has the same name as the class preceded by a **tild ~** symbol. It is called at the end of code or when the object is destroyed or goes out of scope.

Syntax

```
class cl_name{
        ~ cl_name(){} //destructor
}
```

Q3) What is the use of constructor?

Ans.

A constructor is a method that has the same name as a class. And the use of a constructor is to initialize the object when it is created using a **new** keyword.

When an object is created, the variables are initialized chunks of memory and base values if there are any.

Q4) What is the use of destructor?

Ans.

A destructor is a method that has the same name as a class preceding a ~ symbol. The use of a destructor is to deallocate the memory chunks one the code goes out of the scope of the object or deleted using the **delete keyword**.

When the object is deleted the destructor is called and it deallocated all the memory blocks that were created when an object was created.

Q5) What is the order of constructor execution in C++?

Ans.

A constructor is invoked when the object of a class is created. The order in which a constructor is invoked is the same as the hierarchy of the inheritance. This means that first the object of a base class is invoked then the objects of the child class are invoked and so on.

Q6) What is the order of destructor execution in C++?

Ans.

A destructor is invoked in the reverse order as the constructor and is invoked when the object of the class is deleted. The order in which a destructor is invoked is just the opposite of the hierarchy of the inheritance. This means that first the object of child class is destroyed then the objects of the parent class are destroyed and so on.

Q7) Is the default constructor created even if we create any other constructor?

Ans.

Constructors are created by default by the compiler if a programmer doesn't define any constructor explicitly. If the programmer defines a constructor then compiler holds its work and does not define any of it.

EXPERIMENT - 13

Object Oriented Programming Lab

Aim

Write a program to enter any number and find its factorial using constructor.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 13

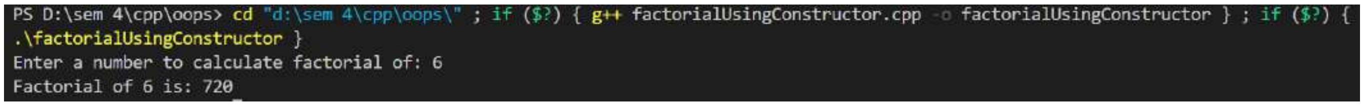
Aim:

Write a program to enter any number and find its factorial using constructor.

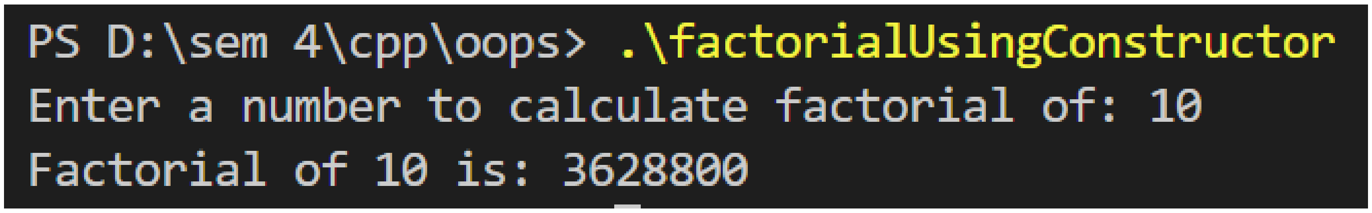
Source Code:

```cpp
#include<iostream>
using namespace std;

class factorial{
    int n, i, f;

    public:
        factorial(){
            cout << "Enter a number to calculate factorial of: ";
            cin>>n;

            f = 1;
            for (i = 1; i <= n; i++) {
                f = f * i;
            }
        }

        void executeFac() {
            cout << "Factorial of " << n << " is: " << f;
        }
};

int main() {

    factorial factObj1;
    factObj1.executeFac();

    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ factorialUsingConstructor.cpp -o factorialUsingConstructor } ; if ($?) { .\factorialUsingConstructor }
Enter a number to calculate factorial of: 6
Factorial of 6 is: 720
```



```
PS D:\sem 4\cpp\oops> .\factorialUsingConstructor
Enter a number to calculate factorial of: 10
Factorial of 10 is: 3628800
```

Viva Questions

Q1) When are copy constructors called in C++?

There are some possible situation when copy constructor called in C++,

- When an object of the class is returned by value.
- When an object of the class is passed (to a function) by value as an argument.
- When an object is constructed based on another object of the same class.
- When the compiler generates a temporary object.

Q2) Why copy constructor takes the parameter as a reference in C++?

A copy constructor is called when an object is passed by value. The copy constructor itself is a function. So if we pass an argument by value in a copy constructor, a call to copy constructor would be made to call copy constructor which becomes a non-terminating chain of calls. Therefore compiler doesn't allow parameters to be passed by value.

Q3) Why copy constructor argument should be const in C++?

There are some important reasons to use const in the copy constructor.

- const keyword avoids accidental changes.
- You would like to be able to create a copy of the const objects. But if you're not passing your argument with a const qualifier, then you can't create copies of const objects.
- You couldn't create copies from temporary reference, because temporary objects are rvalue, and can't be bound to reference to non-const.

EXPERIMENT - 14

Object Oriented Programming Lab

Aim

Write a program to generate a Fibonacci series using Copy Constructor.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 14

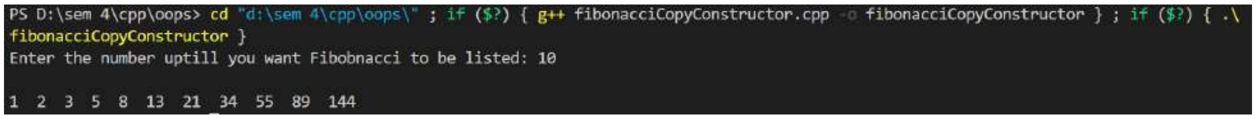
Aim:

Write a program to generate a Fibonacci series using Copy Constructor.

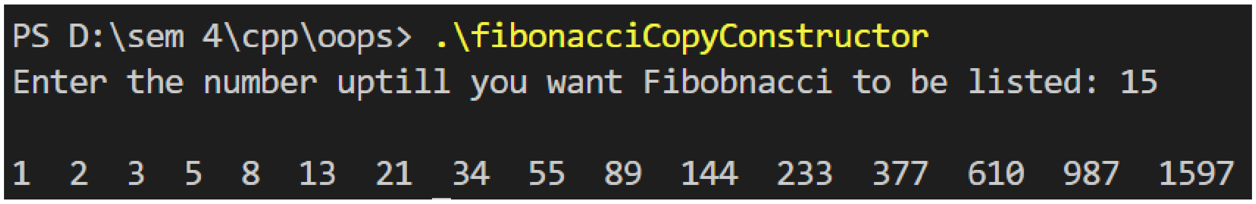
Source Code:

```
#include <iostream>
using namespace std;

class fibonacci{
    private:
        unsigned long int f0, f1, fib;

    public :
        fibonacci(){
            f0 = 0;
            f1 = 1;
            fib = f0 + f1;
        }

        void update(){
            f0 = f1;
            f1 = fib;
            fib = f0 + f1;
        }

        void displayFib(int upto){
            for (int i = 0; i <= upto; i++) {
                cout << fib << "  ";
                update();
            }
        }
}; //end of class construction


int main(){
    fibonacci fibObj;
    int upto;
    cout << "Enter the number uptill you want Fibobnacci to be listed: ";
    cin >> upto;
    cout << endl;
    fibObj.displayFib(upto);
    cout << endl;

    return 0;
}
```

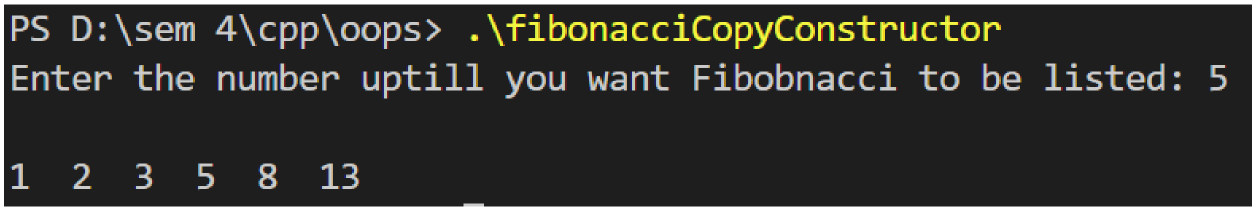
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ fibonacciCopyConstructor.cpp -o fibonacciCopyConstructor } ; if ($?) { .\
fibonacciCopyConstructor }
Enter the number uptill you want Fibobnacci to be listed: 10

1  2  3  5  8  13  21  34  55  89  144
```



```
PS D:\sem 4\cpp\oops> .\fibonacciCopyConstructor
Enter the number uptill you want Fibobnacci to be listed: 15

1  2  3  5  8  13  21  34  55  89  144  233  377  610  987  1597
```



```
PS D:\sem 4\cpp\oops> .\fibonacciCopyConstructor
Enter the number uptill you want Fibobnacci to be listed: 5

1  2  3  5  8  13
```

Viva Questions

Q1) Can a copy constructor accept an object of the same class as a parameter, in place of reference of the object? If No, why not possible?

Ans.

No. It is specified in the definition of the copy constructor itself. It should generate an error if a programmer specifies a copy constructor with a first argument that is an object and not a reference.

Q2) Are Constructors and destructors can declare as const?

Ans.

Constructors and destructors can't be declared as const or volatile. They can, however, be invoked on const or volatile objects.

Q3) Can we make a copy constructor private?

Ans.

Yes, a copy constructor can be made private. When we make a copy constructor private in a class, objects of that class become non-copyable. This is particularly useful when our class has pointers or dynamically allocated resources.

Q4) Can you explain the order of execution in the constructor initialization list?

Ans.

When a class object is created using constructors, the execution order of constructors is:

- Constructors of Virtual base classes are executed, in the order that they appear in the base list.
- Constructors of nonvirtual base classes are executed, in the declaration order.
- Constructors of class members are executed in the declaration order (regardless of their order in the initialization list).
- The body of the constructor is executed.

EXPERIMENT - 15

Object Oriented Programming Lab

Aim

Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the bigger of two entered numbers.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 15

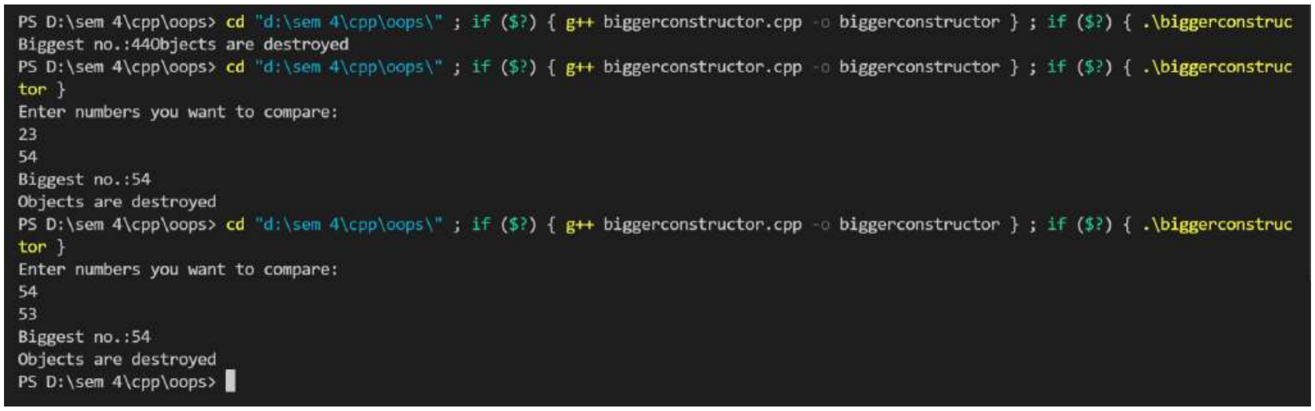
Aim:

Write a program to demonstrate the use of special functions, constructor and destructor in the class template. The program is used to find the bigger of two entered numbers.

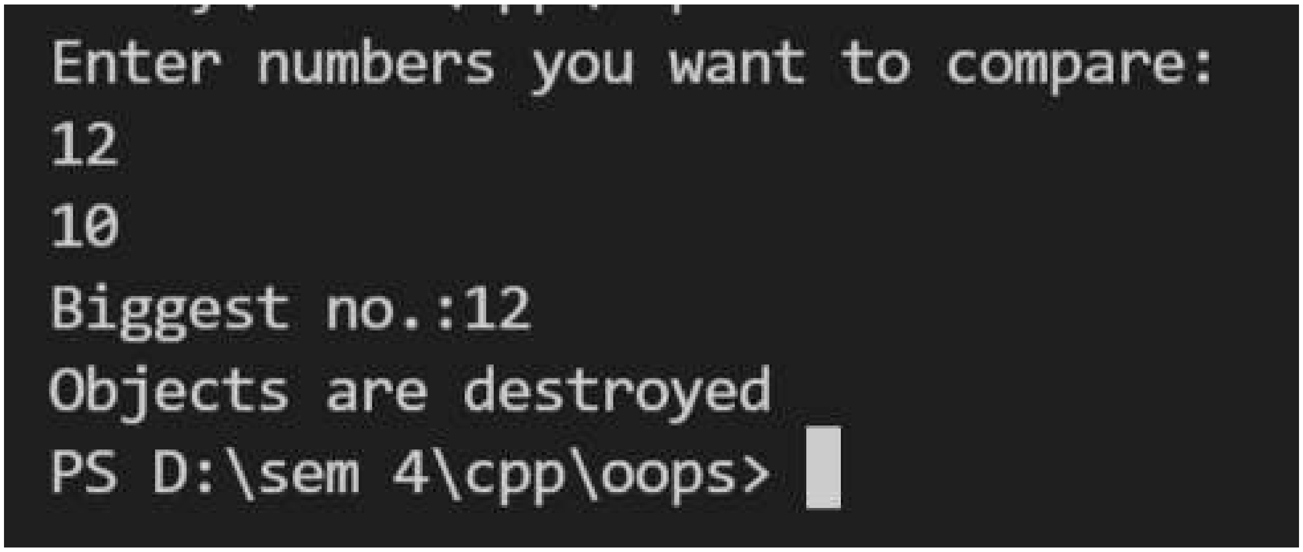
Source Code:

```
#include<iostream>

using namespace std;

class biggest
{
private:
    int a, b;

public:
    void display() //constructor function
    {
        if (a > b)
            cout << "Biggest no.:" << a << endl;
        else
            cout << "Biggest no.:" << b << endl;
    }

    ~biggest() //destructor function
    {
        cout << "Objects are destroyed" << endl;
    }

    biggest(int a, int b) //Bigger of two numbers
    {
       this->a = a;
       this->b = b;
    }
};

int main()
{
    int x, y;
    cout << "Enter numbers you want to compare:" << endl;
    cin >> x >> y;
    biggest bigger (x, y);
    bigger.display();
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ biggerconstructor.cpp -o biggerconstructor } ; if ($?) { .\biggerconstruc
Biggest no.:44Objects are destroyed
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ biggerconstructor.cpp -o biggerconstructor } ; if ($?) { .\biggerconstruc
tor }
Enter numbers you want to compare:
23
54
Biggest no.:54
Objects are destroyed
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ biggerconstructor.cpp -o biggerconstructor } ; if ($?) { .\biggerconstruc
tor }
Enter numbers you want to compare:
54
53
Biggest no.:54
Objects are destroyed
PS D:\sem 4\cpp\oops>
```



```
Enter numbers you want to compare:
12
10
Biggest no.:12
Objects are destroyed
PS D:\sem 4\cpp\oops>
```

Viva Questions

Q1) What is a class constructor?

Ans.

A class constructor is a special member function of a class that is executed whenever we create new objects of that class.

A constructor will have exact same name as the class and it does not have any return type at all, not even void. Constructors can be very useful for setting initial values for certain member variables.

Q2) Are Constructors and destructors can declare as const?

Ans.

Constructors and destructors can’t be declared as const or volatile. They can, however, be invoked on const or volatile objects.

Q3) What is a Destructor?

Ans.

Destructor is a member function which destructs or deletes an object.

EXPERIMENT - 16

Object Oriented Programming Lab

Aim

Create a base class basic_info with data members name, roll no, sex and two member functions getdata and display. Derive a class physical_fit from basic_info which has data members height and weight and member functions getdata and display. Display all the information using object of derived class.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 16

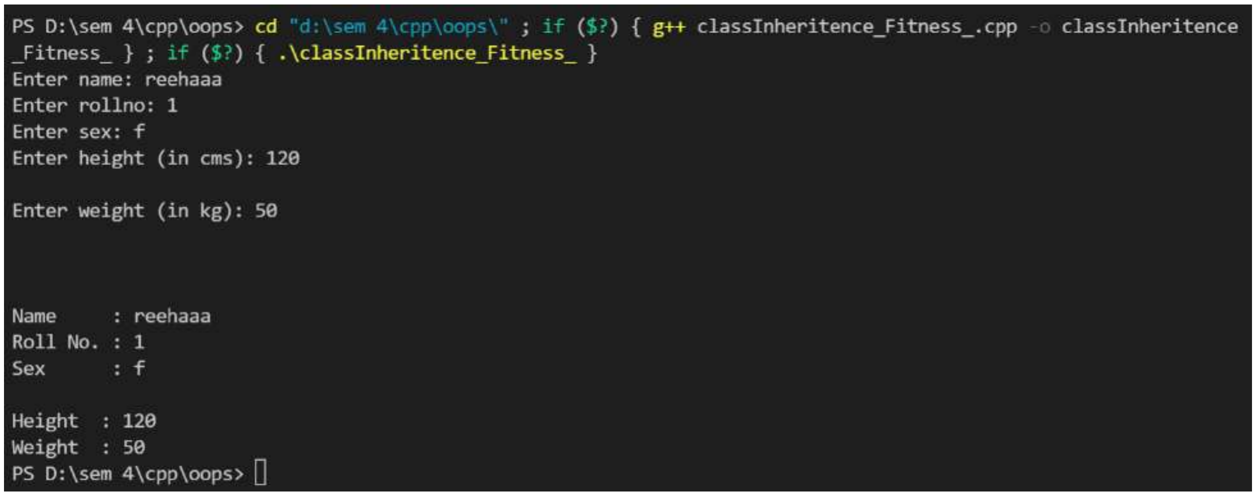
Aim:

Create a base class basic_info with data members name, roll no, sex and two member functions getdata and display. Derive a class physical_fit from basic_info which has data members height and weight and member functions getdata and display. Display all the information using object of derived class.

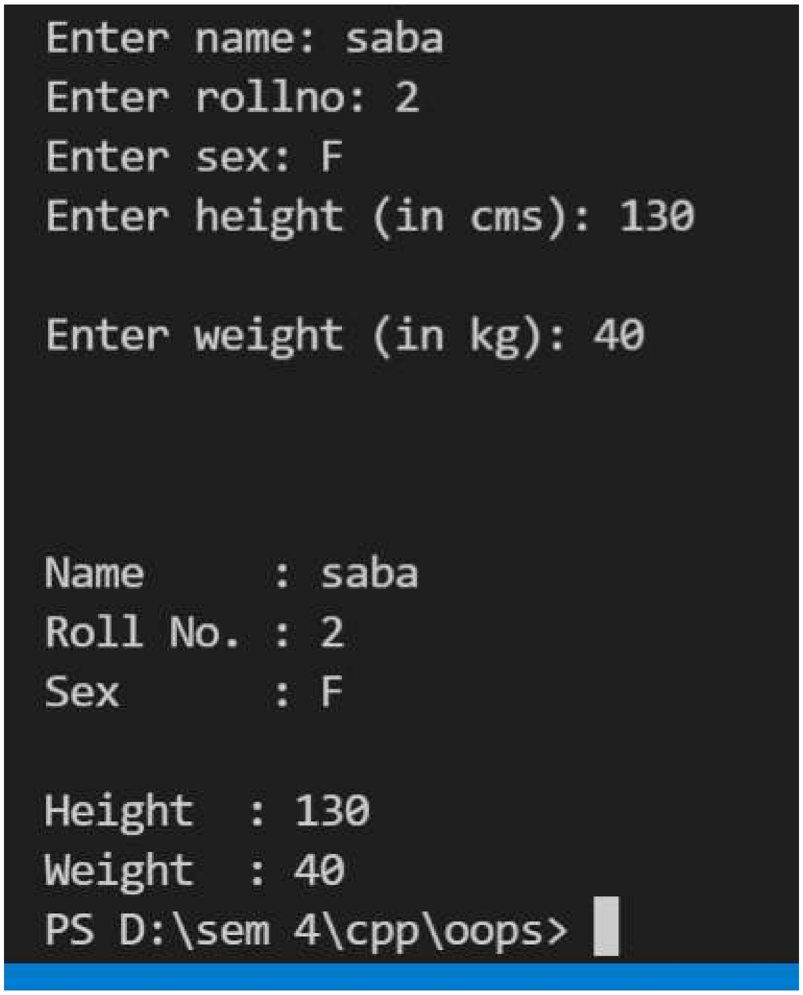
Source Code:

```
#include <iostream>
#include<string>
using namespace std;

class basic_info
{
    string name;
    int rno;
    char sex;

public:
    void getdata();
    void putdata();
};

void basic_info::getdata()
{
    cout << "Enter name: ";
    cin >> name;
    cout << "Enter rollno: ";
    cin >> rno;
    cout << "Enter sex: ";
    cin >> sex;
}

void basic_info::putdata()
{
    cout << "\n\n\n";
    cout << "Name     : " << name << endl;
    cout << "Roll No. : " << rno << endl;
    cout << "Sex      : " << sex << endl;
}

class phy_fit : public basic_info
{
    float ht;
    float wt;

public:
    void input()
    {
        getdata();
        cout << "Enter height (in cms): ";
        cin >> ht;
        cout << "\nEnter weight (in kg): ";
        cin >> wt;
    }
    void display()
    {
        putdata();
        cout << "\nHeight  : " << ht;
        cout << "\nWeight  : " << wt;
    }
};

main()
{
    phy_fit obj;
    obj.input();
    obj.display();

    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ classInheritence_Fitness_.cpp -o classInheritence
_Fitness_ } ; if ($?) { .\classInheritence_Fitness_ }
Enter name: reehaaa
Enter rollno: 1
Enter sex: f
Enter height (in cms): 120

Enter weight (in kg): 50



Name     : reehaaa
Roll No. : 1
Sex      : f

Height  : 120
Weight  : 50
PS D:\sem 4\cpp\oops> 
```



```
Enter name: saba
Enter rollno: 2
Enter sex: F
Enter height (in cms): 130

Enter weight (in kg): 40



Name     : saba
Roll No. : 2
Sex      : F

Height  : 130
Weight  : 40
PS D:\sem 4\cpp\oops> 
```

Viva Questions

Q1) What is inheritance?

Ans.

Inheritance is one of the feature of Object Oriented Programming System(OOPs), it allows the child class to acquire the properties (the data members) and functionality (the member functions) of parent class.

Q2) What is child class?

Ans.

A class that inherits another class is known as child class, it is also known as derived class or subclass.

Q3) What is parent class?

Ans.

The class that is being inherited by other class is known as parent class, super class or base class.

Q4) What are the advantages of using inheritance in C++ Programming?

Ans.

The main advantages of inheritance are code reusability and readability. When child class inherits the properties and functionality of parent class, we need not to write the same code again in child class. This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable.

EXPERIMENT - 17

Object Oriented Programming Lab

Aim

Create class first with data members book no, book name and member function getdata and putdata. Create a class second with data members author name, publisher and members getdata and showdata. Derive a class third from first and second with data member no of pages and year of publication. Display all these information using array of objects of third class.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 17

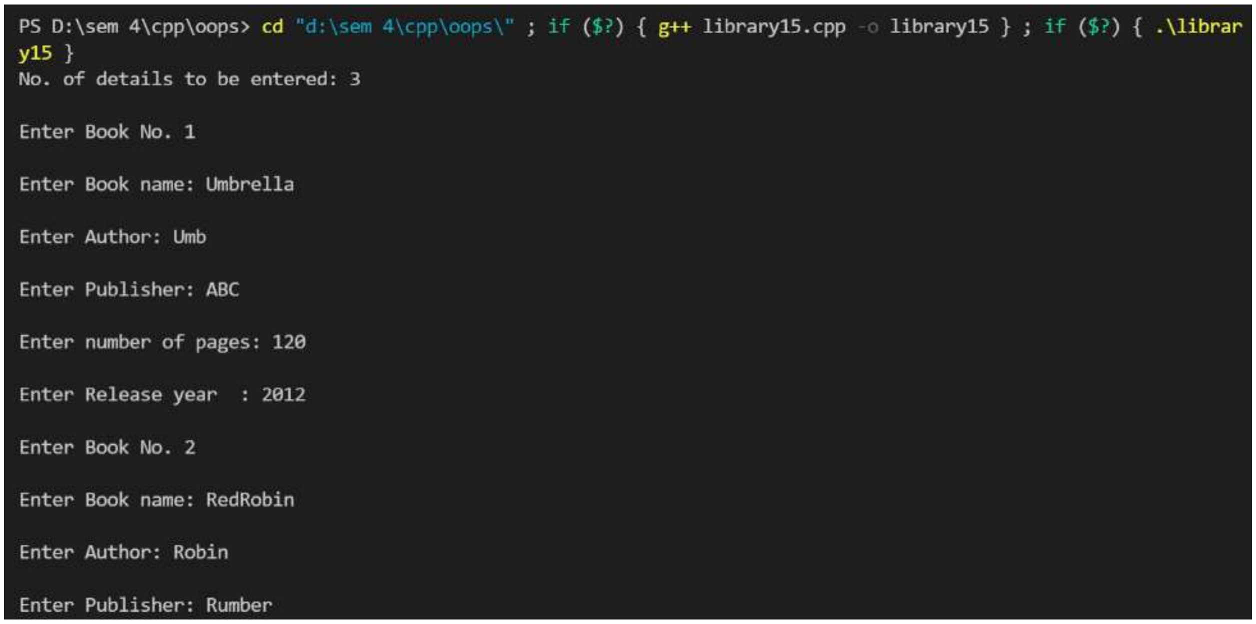
Aim:

Create class first with data members book no, book name and member function getdata and putdata. Create a class second with data members author name, publisher and members getdata and showdata. Derive a class third from first and second with data member no of pages and year of publication. Display all these information using array of objects of third class.

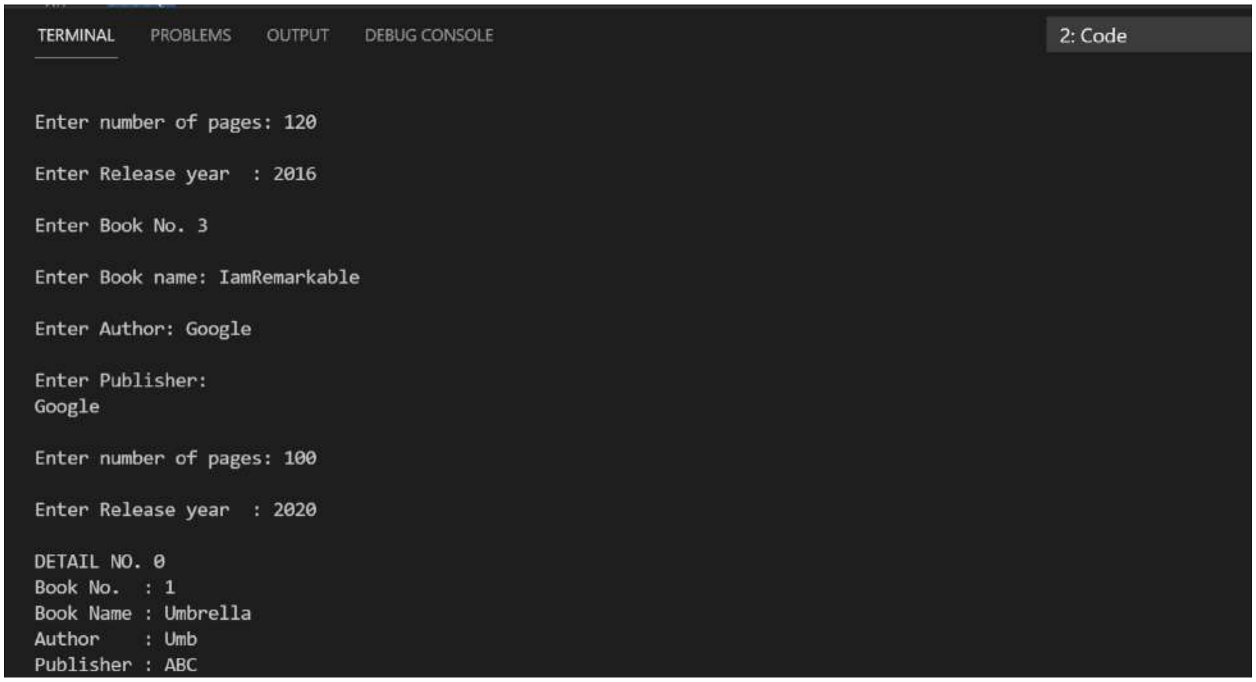
Source Code:

```
#include <iostream>
#include <string>
using namespace std;

class first
{
    int bno;
    char bname[50];

public:
    void input()
    {
        cout << "\nEnter Book No. ";
        cin >> bno;
        cout << "\nEnter Book name: ";
        cin >> bname;
    }
    void output()
    {
        cout << "\nBook No.  : " << bno;
        cout << "\nBook Name : " << bname;
    }
};
class second
{
    char author[25];
    char publisher[25];

public:
    void indata()
    {
        cout << "\nEnter Author: ";
        cin >> author;
        cout << "\nEnter Publisher: ";
        cin >> publisher;
    }
    void outdata()
    {
        cout << "\nAuthor    : " << author;
        cout << "\nPublisher : " << publisher;
    }
};
class third : public first, public second
{
    int pgn;
    int yr;

public:
    void in()
    {
        input();
        indata();
        cout << "\nEnter number of pages: ";
        cin >> pgn;
        cout << "\nEnter Release year  : ";
        cin >> yr;
    }
    void out()
    {
        output();
        outdata();
        cout << "\nNumber of pages: " << pgn;
        cout << "\nRelease Year   : " << yr;
    }
};

main()
{
    third t[5];
    int i, n;
    cout << "No. of details to be entered: ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        t[i].in();
    }
    for (i = 0; i < n; i++)
    {
        cout << "\nDETAIL NO. " << i;
        t[i].out();
    }
    return 0;
}
```

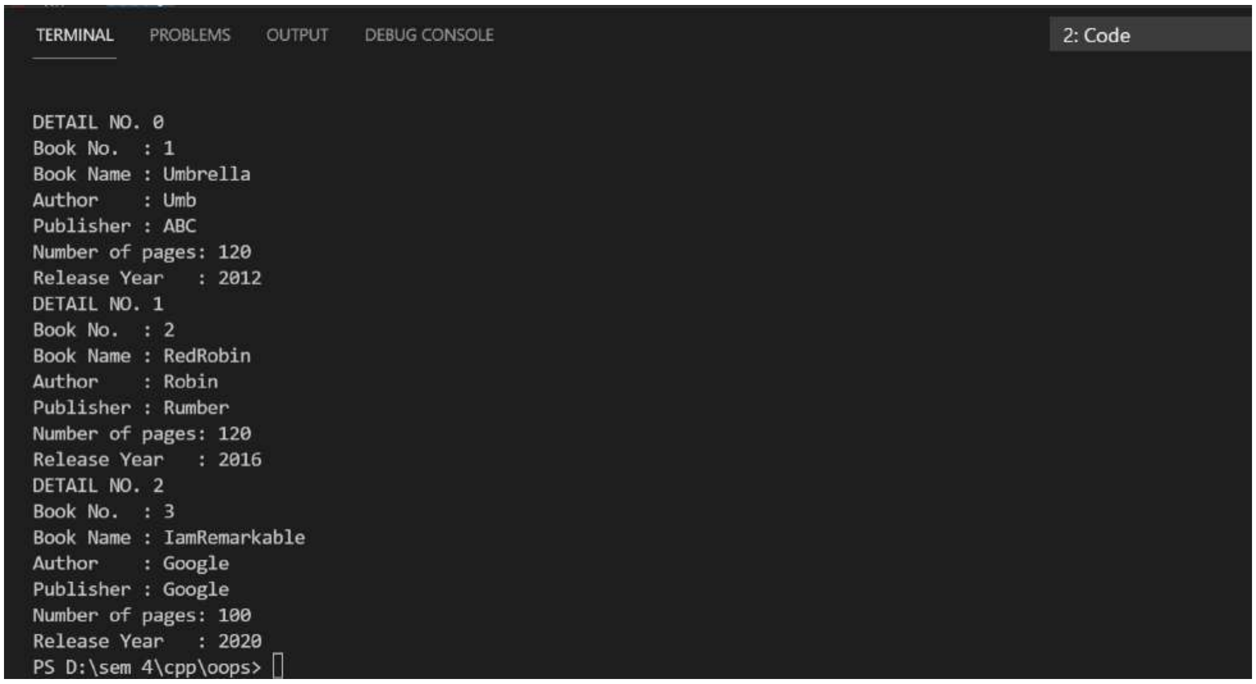
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ library15.cpp -o library15 } ; if ($?) { .\librar
y15 }
No. of details to be entered: 3

Enter Book No. 1

Enter Book name: Umbrella

Enter Author: Umb

Enter Publisher: ABC

Enter number of pages: 120

Enter Release year  : 2012

Enter Book No. 2

Enter Book name: RedRobin

Enter Author: Robin

Enter Publisher: Rumber
```



```
Enter number of pages: 120

Enter Release year  : 2016

Enter Book No. 3

Enter Book name: IamRemarkable

Enter Author: Google

Enter Publisher:
Google

Enter number of pages: 100

Enter Release year  : 2020

DETAIL NO. 0
Book No.   : 1
Book Name  : Umbrella
Author     : Umb
Publisher  : ABC
```



```
DETAIL NO. 0
Book No.  : 1
Book Name : Umbrella
Author    : Umb
Publisher : ABC
Number of pages: 120
Release Year   : 2012
DETAIL NO. 1
Book No.  : 2
Book Name : RedRobin
Author    : Robin
Publisher : Rumber
Number of pages: 120
Release Year   : 2016
DETAIL NO. 2
Book No.  : 3
Book Name : IamRemarkable
Author    : Google
Publisher : Google
Number of pages: 100
Release Year   : 2020
PS D:\sem 4\cpp\oops>
```

Viva Questions

Q1) What is inheritance?

Ans.

Inheritance is one of the feature of Object Oriented Programming System(OOPs), it allows the child class to acquire the properties (the data members) and functionality (the member functions) of parent class.

Q2) What is child class?

Ans.

A class that inherits another class is known as child class, it is also known as derived class or subclass.

Q3) What is parent class?

Ans.

The class that is being inherited by other class is known as parent class, super class or base class.

Q4) What are the advantages of using inheritance in C++ Programming?

Ans.

The main advantages of inheritance are code reusability and readability. When child class inherits the properties and functionality of parent class, we need not to write the same code again in child class. This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable.

Q5) What are advantages and disadvantages of inheritance?

Ans.

Disadvantages: -

- Inherited functions work slower than normal function as there is indirection.
- Improper use of inheritance may lead to wrong solutions.
- Often, data members in the base class are left unused which may lead to memory wastage.
- Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes.

Advantages:

- Inheritance promotes reusability. When a class inherits or derives another class, it can access all the functionality of inherited class.
- Reusability enhanced reliability. The base class code will be already tested and debugged.
- As the existing code is reused, it leads to less development and maintenance costs.

- Inheritance makes the sub classes follow a standard interface.
- Inheritance helps to reduce code redundancy and supports code extensibility.
- Inheritance facilitates creation of class libraries.

EXPERIMENT - 18

Object Oriented Programming Lab

Aim

Write a program to read the class object of student info such as name, age, sex, height and weight from the keyboard and to store them on a specified file using read () and write () functions. Again, the same file is opened for reading and displaying the contents of the file on the screen.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 18

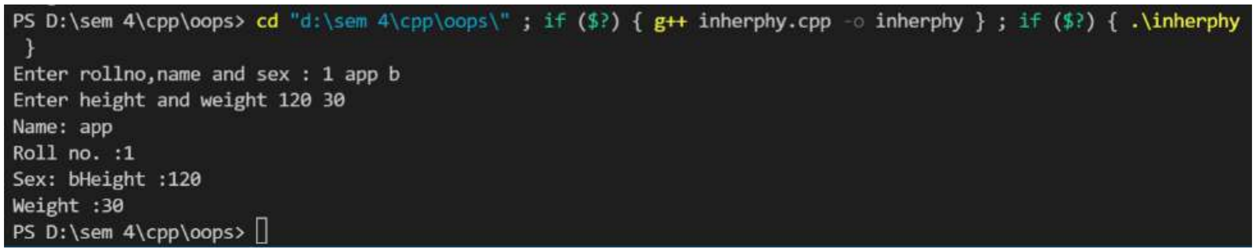
Aim:

Write a program to read the class object of student info such as name, age, sex, height and weight from the keyboard and to store them on a specified file using read () and write () functions. Again, the same file is opened for reading and displaying the contents of the file on the screen.

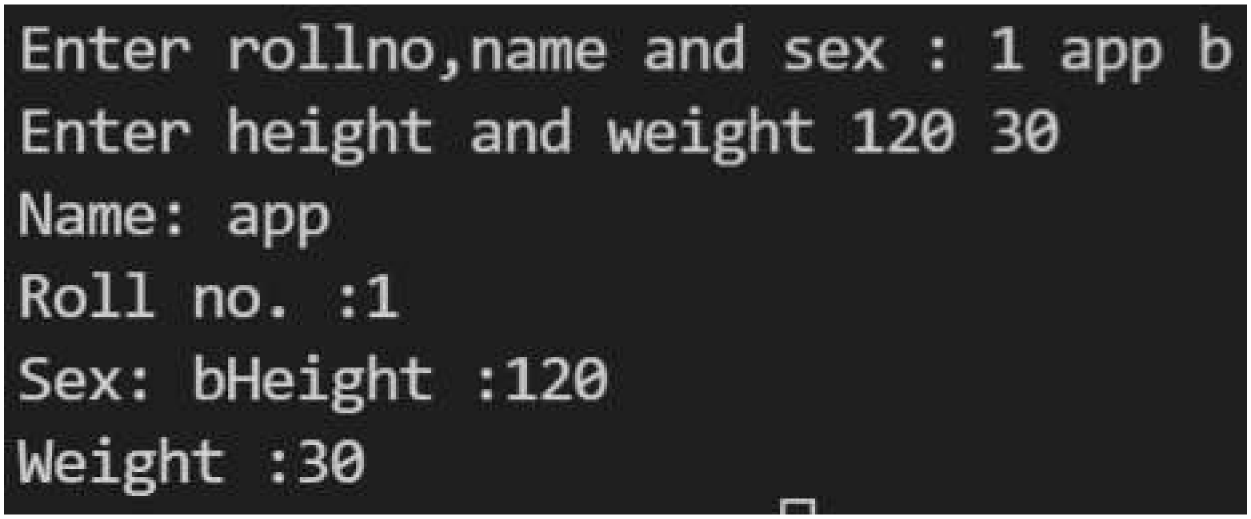
Source Code:

```
#include <iostream>

using namespace std;

class basic_info
{
    char name[10], sex;
    int rollno;

public:
    void getdata()
    {
        cout << "Enter rollno,name and sex : ";
        cin >> rollno >> name >> sex;
    }
    void display()
    {
        cout << "Name: " << name << "\nRoll no. :" << rollno << "\nSex: " << sex;
    }
};

class physical_fit : public basic_info
{
    float height, weight;

public:
    void getdata()
    {
        basic_info::getdata();
        cout << "Enter height and weight ";
        cin >> height >> weight;
    }
    void display()
    {
        basic_info::display();
        cout << "Height :" << height;
        cout << "\nWeight :" << weight;
    }
};

int main()
{
    physical_fit p;
    p.getdata();
    p.display();
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ inherphy.cpp -o inherphy } ; if ($?) { .\inherphy
 }
Enter rollno,name and sex : 1 app b
Enter height and weight 120 30
Name: app
Roll no. :1
Sex: bHeight :120
Weight :30
PS D:\sem 4\cpp\oops> 
```



```
Enter rollno,name and sex : 1 app b
Enter height and weight 120 30
Name: app
Roll no. :1
Sex: bHeight :120
Weight :30
```

Viva Questions

Q1) What is inheritance?

Ans.

Inheritance is one of the feature of Object Oriented Programming System(OOPs), it allows the child class to acquire the properties (the data members) and functionality (the member functions) of parent class.

Q2) What is child class?

Ans.

A class that inherits another class is known as child class, it is also known as derived class or subclass.

Q3) What is parent class?

Ans.

The class that is being inherited by other class is known as parent class, super class or base class.

Q4) What are the advantages of using inheritance in C++ Programming?

Ans.

The main advantages of inheritance are code reusability and readability. When child class inherits the properties and functionality of parent class, we need not to write the same code again in child class. This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable.

Q5) What are advantages and disadvantages of inheritance?

Ans.

Disadvantages: -

- Inherited functions work slower than normal function as there is indirection.
- Improper use of inheritance may lead to wrong solutions.
- Often, data members in the base class are left unused which may lead to memory wastage.
- Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes.

Advantages:

- Inheritance promotes reusability. When a class inherits or derives another class, it can access all the functionality of inherited class.
- Reusability enhanced reliability. The base class code will be already tested and debugged.
- As the existing code is reused, it leads to less development and maintenance costs.

- Inheritance makes the sub classes follow a standard interface.
- Inheritance helps to reduce code redundancy and supports code extensibility.
- Inheritance facilitates creation of class libraries.

EXPERIMENT - 19

Object Oriented Programming Lab

Aim

Create a class called LIST with two pure virtual function store () and retrieve (). To store a value call store and to retrieve call retrieve function. Derive two classes stack and queue from it and override store and retrieve.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 19

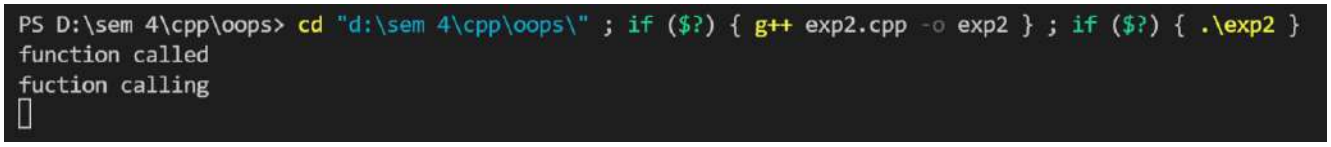
Aim:

Create a class called LIST with two pure virtual function store () and retrieve (). To store a value call store and to retrieve call retrieve function. Derive two classes stack and queue from it and override store and retrieve.

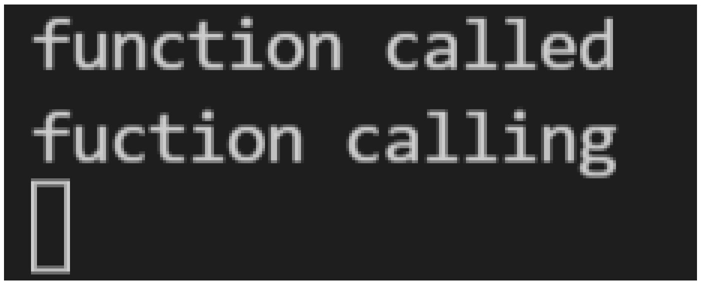
Source Code:

```
#include <iostream>
#include <conio.h>

using namespace std;

class LIST
{
public:
    virtual void store() = 0;
    virtual void retrieve() = 0;
};

class B : public LIST
{
public:
    void store()
    {
        cout << "function called" << endl;
    }
    void retrieve()
    {
        cout << "fuction calling" << endl;
    }
};

int main()
{
    B obj;
    obj.store();
    obj.retrieve();
    getch();
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ exp2.cpp -o exp2 } ; if ($?) { .\exp2 }
function called
fuction calling
```



```
function called
fuction calling
```

Viva Questions

Q1) What is a virtual function?

Ans.

A virtual function is a member function in the base class that we expect to redefine in derived classes.

Basically, a virtual function is used in the base class in order to ensure that the function is overridden. This especially applies to cases where a pointer of base class points to an object of a derived class.

Q2) What is child class?

Ans.

A class that inherits another class is known as child class, it is also known as derived class or subclass.

Q3) What is parent class?

Ans.

The class that is being inherited by other class is known as parent class, super class or base class.

Q4) What are the advantages of using inheritance in C++ Programming?

Ans.

The main advantages of inheritance are code reusability and readability. When child class inherits the properties and functionality of parent class, we need not to write the same code again in child class. This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable.

Q5) What are advantages and disadvantages of inheritance?

Ans.

Disadvantages: -

- Inherited functions work slower than normal function as there is indirection.
- Improper use of inheritance may lead to wrong solutions.
- Often, data members in the base class are left unused which may lead to memory wastage.
- Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes.

Advantages:

- Inheritance promotes reusability. When a class inherits or derives another class, it can access all the functionality of inherited class.

- Reusability enhanced reliability. The base class code will be already tested and debugged.
- As the existing code is reused, it leads to less development and maintenance costs.
- Inheritance makes the sub classes follow a standard interface.
- Inheritance helps to reduce code redundancy and supports code extensibility.
- Inheritance facilitates creation of class libraries.

EXPERIMENT - 20

Object Oriented Programming Lab

Aim

Create a base class called SHAPE. Use this class to store two double type values. Derive two specific classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata to initialize base class data members and another member function display to compute and display the area of figures. Make display a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes design a program that will accept driven of a TRIANGLE or RECTANGLE interactively and display the area.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 20

Aim:

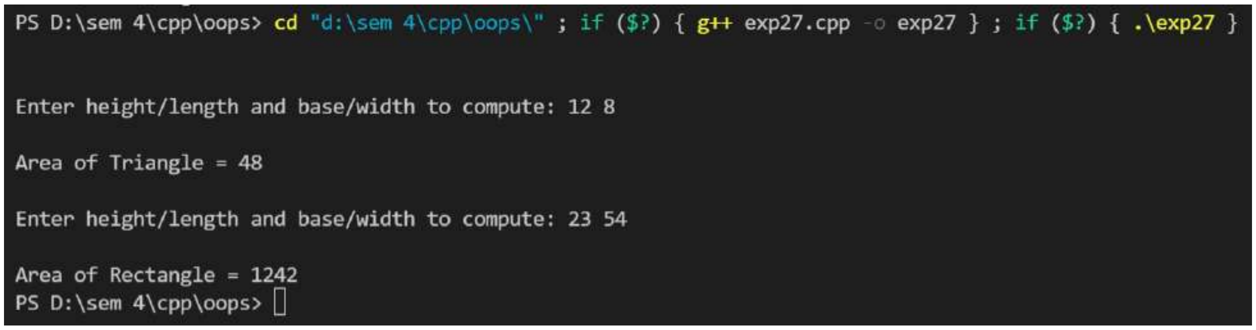
Create a base class called SHAPE. Use this class to store two double type values. Derive two specific classes called TRIANGLE and RECTANGLE from the base class. Add to the base class, a member function getdata to initialize base class data members and another member function display to compute and display the area of figures. Make display a virtual function and redefine this function in the derived classes to suit their requirements. Using these three classes design a program that will accept driven of a TRIANGLE or RECTANGLE interactively and display the area.

Source Code:

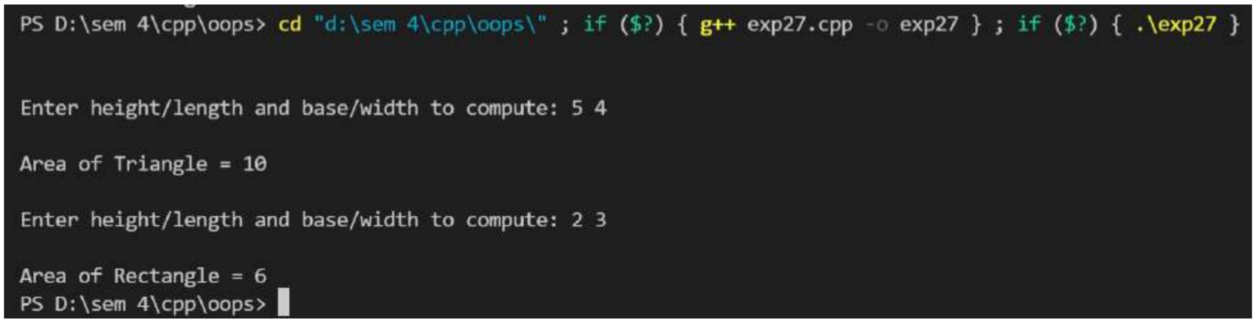
```
#include <iostream>

using namespace std;

class Shape
{
public:
    double height, base;

    Shape()
    {
        height = 0;
        base = 0;
    }

    void get_data()
    {
        cout << "\n\nEnter height/length and base/width to compute: ";
        cin >> height >> base;
    }

    virtual void display_area() {}
};

class Triangle : public Shape
```

```
{
public:
    void display_area()
    {
        cout << "\nArea of Triangle = " << (height * base) / 2;
    }
};

class Rectangle : public Shape
{
public:
    void display_area()
    {
        cout << "\nArea of Rectangle = " << height * base;
    }
};
int main()
{

    Shape *s;
    Triangle t;
    t.get_data();
    s = &t;
    s->display_area();
    Rectangle r;
    r.get_data();
    s = &r;
    s->display_area();
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ exp27.cpp -o exp27 } ; if ($?) { .\exp27 }


Enter height/length and base/width to compute: 12 8

Area of Triangle = 48

Enter height/length and base/width to compute: 23 54

Area of Rectangle = 1242
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ exp27.cpp -o exp27 } ; if ($?) { .\exp27 }


Enter height/length and base/width to compute: 5 4

Area of Triangle = 10

Enter height/length and base/width to compute: 2 3

Area of Rectangle = 6
PS D:\sem 4\cpp\oops> 
```

Viva Questions

Q1) What is inheritance?

Ans.

Inheritance is one of the feature of Object Oriented Programming System(OOPs), it allows the child class to acquire the properties (the data members) and functionality (the member functions) of parent class.

Q2) What is a virtual function?

Ans.

A virtual function is a member function in the base class that we expect to redefine in derived classes.

Basically, a virtual function is used in the base class in order to ensure that the function is overridden. This especially applies to cases where a pointer of base class points to an object of a derived class.

Q3) What is child class?

Ans.

A class that inherits another class is known as child class, it is also known as derived class or subclass.

Q4) What is parent class?

Ans.

The class that is being inherited by other class is known as parent class, super class or base class.

Q5) What are the advantages of using inheritance in C++ Programming?

Ans.

The main advantages of inheritance are code reusability and readability. When child class inherits the properties and functionality of parent class, we need not to write the same code again in child class. This makes it easier to reuse the code, makes us write the less code and the code becomes much more readable.

EXPERIMENT - 21

Object Oriented Programming Lab

Aim

Design three classes STUDENT, EXAM, and RESULT. The STUDENT class has data members such as roll no, name. Create a class EXAM by inheriting the STUDENT class. The EXAM class adds data members representing the marks scored in six subjects. Derive the RESULT from the EXAM class and has its own data members such as total marks. WAP to model this relationship.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 21

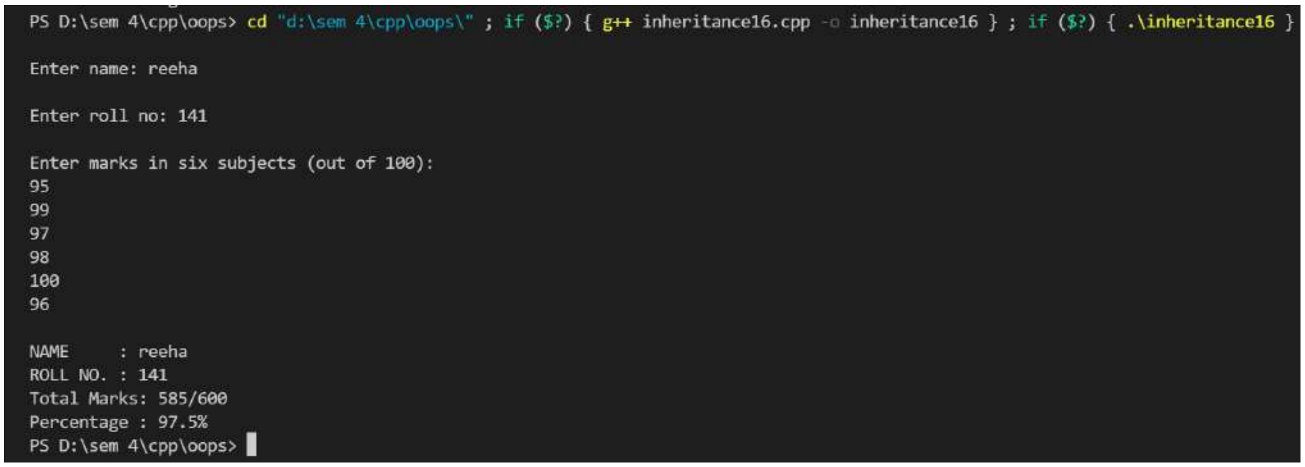
Aim:

Design three classes STUDENT, EXAM, and RESULT. The STUDENT class has data members such as roll no, name. Create a class EXAM by inheriting the STUDENT class. The EXAM class adds data members representing the marks scored in six subjects. Derive the RESULT from the EXAM class and has its own data members such as total marks. WAP to model this relationship.

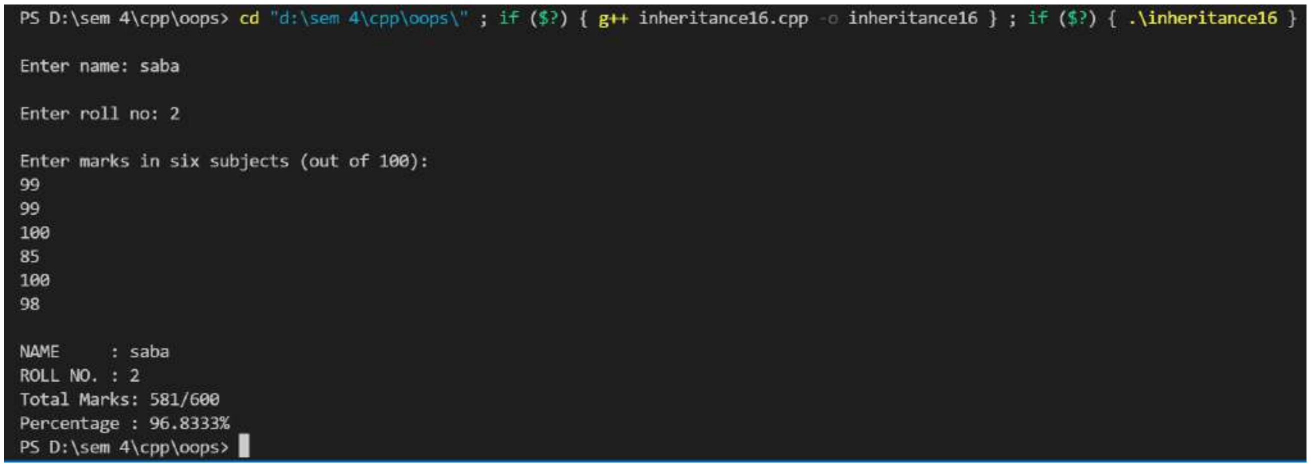
Source Code:

```
#include <iostream>
#include <string>
using namespace std;

class student
{
    string name;
    int rollNo;

public:
    void input()
    {
        cout << "\nEnter name: ";
        cin >> name;
        cout << "\nEnter roll no: ";
        cin >> rollNo;
    }
    void output()
    {
        cout << "\nNAME     : " << name;
        cout << "\nROLL NO. : " << rollNo;
    }
};

class exam : public student
{
    float marks[6];

public:
    void enterData();
    float add();
};

void exam::enterData()
{
    int i;
    input();
    cout << "\nEnter marks in six subjects (out of 100):\n";
    for (i = 0; i < 6; i++)
    {
        cin >> marks[i];
    }
}

float exam::add()
{
    int i = 0;
    float sum = 0;
    for (i = 0; i < 6; i++)
    {
        sum = sum + marks[i];
    }
    return sum;
}

class result : public exam
{
    float totalMarks;
    float percentage;

public:
    void display()
    {
        totalMarks = add();
        percentage = totalMarks / 6;
        output();
        cout << "\nTotal Marks: " << totalMarks << "/600";
        cout << "\nPercentage : " << percentage << "%";
    }
};

main()
{
    result r;
    r.enterData();
    r.display();
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ inheritance16.cpp -o inheritance16 } ; if ($?) { .\inheritance16 }

Enter name: reeha

Enter roll no: 141

Enter marks in six subjects (out of 100):
95
99
97
98
100
96

NAME     : reeha
ROLL NO. : 141
Total Marks: 585/600
Percentage : 97.5%
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ inheritance16.cpp -o inheritance16 } ; if ($?) { .\inheritance16 }

Enter name: saba

Enter roll no: 2

Enter marks in six subjects (out of 100):
99
99
100
85
100
98

NAME     : saba
ROLL NO. : 2
Total Marks: 581/600
Percentage : 96.8333%
PS D:\sem 4\cpp\oops> 
```

Viva Questions

Q1) What is inheritance?

Ans.

Inheritance is one of the feature of Object Oriented Programming System(OOPs), it allows the child class to acquire the properties (the data members) and functionality (the member functions) of parent class.

Q2) What are advantages of inheritance?

Ans.

Code reusability: Now you can reuse the members of your parent class. So, there is no need to define the member again. So less code is required in the class.

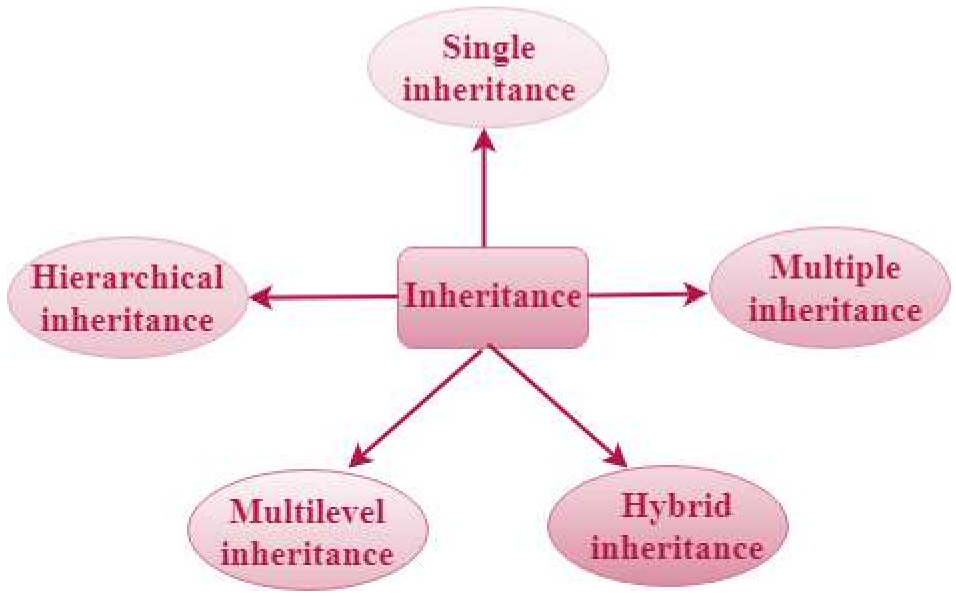
Q3) What are various types of inheritance?

Ans.

Various types of inheritance in C++ are:

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance

- Hybrid inheritance



Q4) How to make private member inheritable?

Ans.

The private member is not inheritable. If we modify the visibility mode by making it public, but this takes away the advantage of data hiding.

C++ introduces a third visibility modifier, i.e., protected. The member which is declared as protected will be accessible to all the member functions within the class as well as the class immediately derived from it.

EXPERIMENT - 22

Object Oriented Programming Lab

Aim

Implement a class string containing the following functions:

a) Overload + operator to carry out the concatenation of strings.
b) Overload = operator to carry out string copy.
c) Overload <= operator to carry out the comparison of strings.
d) Function to display the length of string.
e) Function tolower() to convert upper case to lower case.
f) Function toupper() to convert lower case letters to upper case.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 22

Aim:

Implement a class string containing the following functions:

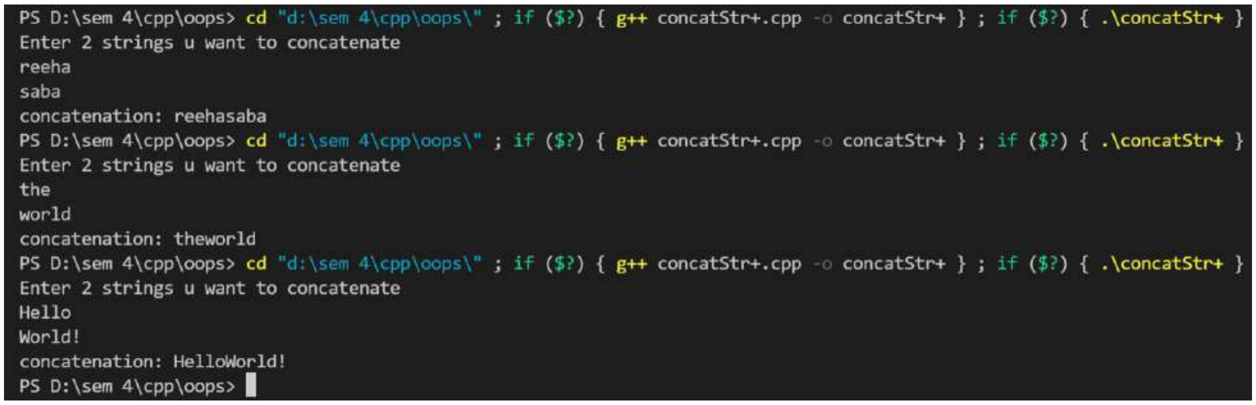
a. Overload + operator to carry out the concatenation of strings.
b. Overload = operator to carry out string copy.
c. Overload <= operator to carry out the comparison of strings.
d. Function to display the length of string.
e. Function tolower() to convert upper case to lower case.
f. Function toupper() to convert lower case letters to upper case.

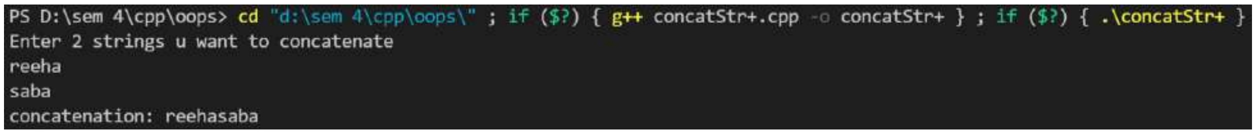
Overload + operator to carry out the concatenation of strings.

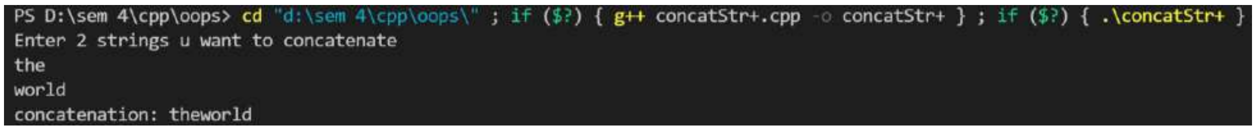
Source Code:

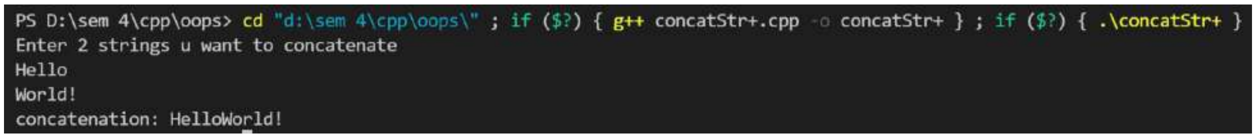
```
#include <iostream>
#include <cstring>

using namespace std;

// concatenating 2 strings
class concatString{
    public:
        char str[100]; // class object

        concatString(){} // no parameter constructor

        // initalising class variable
        concatString(char s[]){
            strcpy(this->str, s);
        }

        // overloading operator+ for concatenation
        concatString operator+(concatString& s2){
            concatString s3;
            strcat(this->str, s2.str);
            strcpy(s3.str, this->str);
            return s3;
        }
};

int main(){
    char s1[100], s2[100];

    cout << "Enter 2 strings u want to concatenate" << endl;
    cin >> s1 >> s2;

    concatString a1(s1);
    concatString a2(s2);
    concatString a3;

    a3 = a1 + a2;

    cout << "concatenation: " << a3.str;

    return 0;
}
```

Output:


```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ concatStr+.cpp -o concatStr+ } ; if ($?) { .\concatStr+ }
Enter 2 strings u want to concatenate
reeha
saba
concatenation: reehasaba
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ concatStr+.cpp -o concatStr+ } ; if ($?) { .\concatStr+ }
Enter 2 strings u want to concatenate
the
world
concatenation: theworld
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ concatStr+.cpp -o concatStr+ } ; if ($?) { .\concatStr+ }
Enter 2 strings u want to concatenate
Hello
World!
concatenation: HelloWorld!
PS D:\sem 4\cpp\oops>
```


```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ concatStr+.cpp -o concatStr+ } ; if ($?) { .\concatStr+ }
Enter 2 strings u want to concatenate
reeha
saba
concatenation: reehasaba
```


```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ concatStr+.cpp -o concatStr+ } ; if ($?) { .\concatStr+ }
Enter 2 strings u want to concatenate
the
world
concatenation: theworld
```

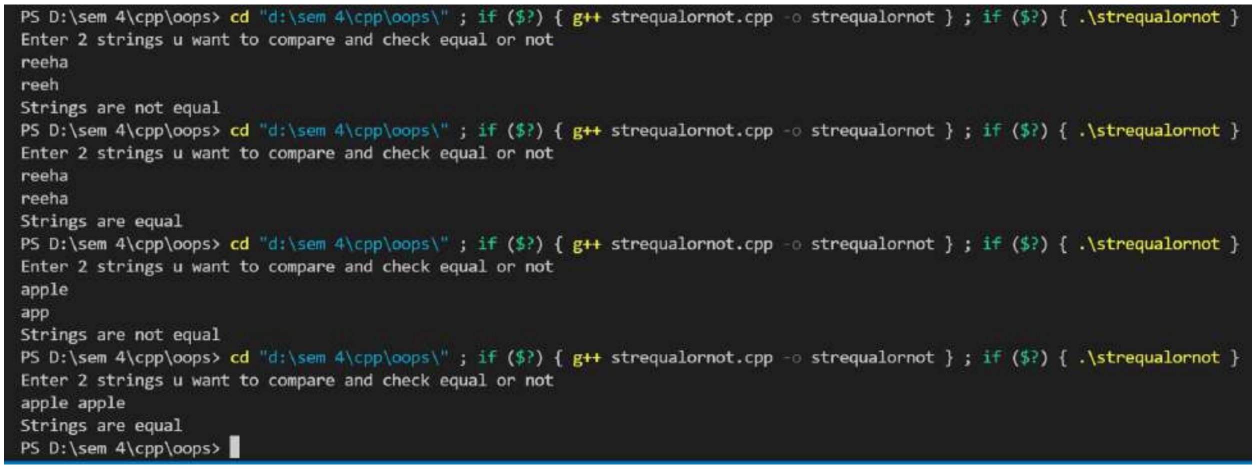

```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ concatStr+.cpp -o concatStr+ } ; if ($?) { .\concatStr+ }
Enter 2 strings u want to concatenate
Hello
World!
concatenation: HelloWorld!
```

Overload = operator to carry out string copy

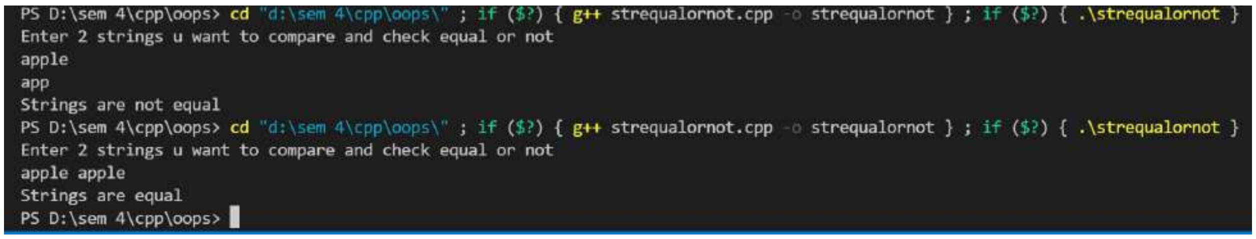
Source Code:

```
#include <iostream>
#include <cstring>

using namespace std;

class equalString{
    public:
        char str[25]; // Classes object of string

        equalString(){} // no parameter constructor

        // Parametrized Constructor
        equalString(char s[]){
            strcpy(this->str, s); // Initialize the string to class object
        }

        bool operator==(equalString s2)
        {
            if (strcmp(str, s2.str) == 0)
                return true;
            else
                return false;
        }
};

int main(){
    char s1[100], s2[100];

    cout << "Enter 2 strings u want to compare and check equal or not" << endl;
    cin >> s1 >> s2;

    equalString a1(s1);
    equalString a2(s2);

    if (a1 == a2) {
        cout << "Strings are equal" << endl;
    }
    else{
        cout << "Strings are not equal" << endl;
    }
    return 0;
}
```

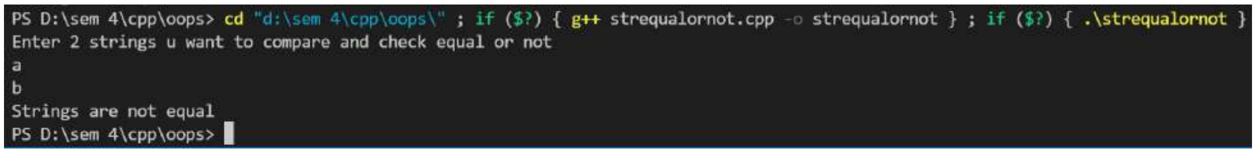
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strequalornot.cpp -o strequalornot } ; if ($?) { .\strequalornot }
Enter 2 strings u want to compare and check equal or not
reeha
reeh
Strings are not equal
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strequalornot.cpp -o strequalornot } ; if ($?) { .\strequalornot }
Enter 2 strings u want to compare and check equal or not
reeha
reeha
Strings are equal
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strequalornot.cpp -o strequalornot } ; if ($?) { .\strequalornot }
Enter 2 strings u want to compare and check equal or not
apple
app
Strings are not equal
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strequalornot.cpp -o strequalornot } ; if ($?) { .\strequalornot }
Enter 2 strings u want to compare and check equal or not
apple apple
Strings are equal
PS D:\sem 4\cpp\oops>
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strequalornot.cpp -o strequalornot } ; if ($?) { .\strequalornot }
Enter 2 strings u want to compare and check equal or not
apple
app
Strings are not equal
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strequalornot.cpp -o strequalornot } ; if ($?) { .\strequalornot }
Enter 2 strings u want to compare and check equal or not
apple apple
Strings are equal
PS D:\sem 4\cpp\oops>
```



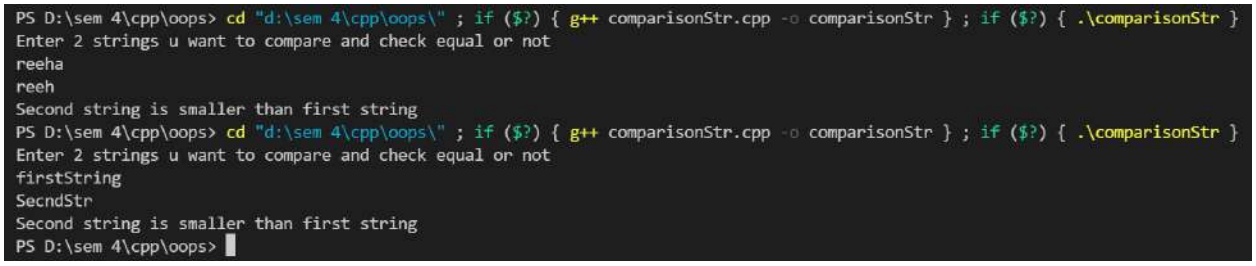
```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strequalornot.cpp -o strequalornot } ; if ($?) { .\strequalornot }
Enter 2 strings u want to compare and check equal or not
a
b
Strings are not equal
PS D:\sem 4\cpp\oops>
```

Overload <= operator to carry out the comparison of strings.

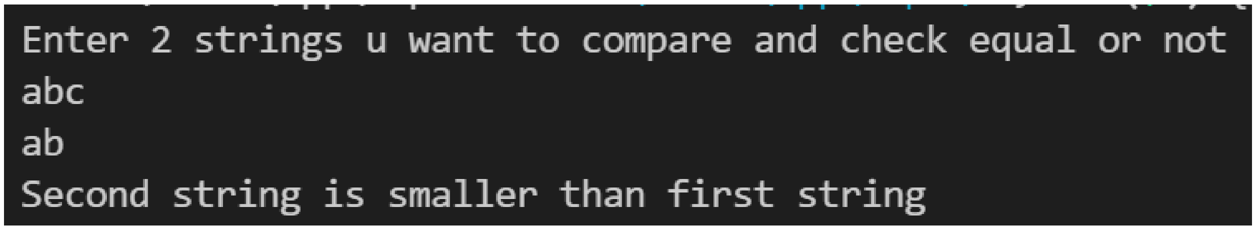
Source Code:

```
#include <cstring>
#include <iostream>
#include <string.h>

using namespace std;

class equalString{
    public:
        char str[25]; // Classes object of string

        equalString(){} // no parameter constructor

        // Parametrized Constructor
        equalString(char s[]){
            strcpy(this->str, s); // Initialize the string to class object
        }

        bool operator<=(equalString s2)
        {
            if (strlen(str) <= strlen(s2.str))
                return true;
            else
                return false;
        }
};

int main(){
    char s1[100], s2[100];

    cout << "Enter 2 strings u want to compare and check equal or not" << endl;
    cin >> s1 >> s2;

    equalString a1(s1);
    equalString a2(s2);

    if (a1 <= a2) {
        cout << "First string is smaller than or equal to second " << endl;
    }
    else{
        cout << "Second string is smaller than first string" << endl;
    }
    return 0;
}
```

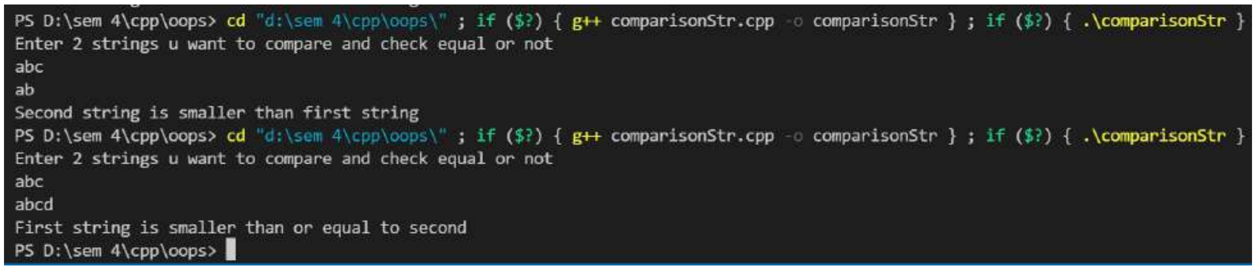
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ comparisonStr.cpp -o comparisonStr } ; if ($?) { .\comparisonStr }
Enter 2 strings u want to compare and check equal or not
reeha
reeh
Second string is smaller than first string
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ comparisonStr.cpp -o comparisonStr } ; if ($?) { .\comparisonStr }
Enter 2 strings u want to compare and check equal or not
firstString
SecndStr
Second string is smaller than first string
PS D:\sem 4\cpp\oops>
```



```
Enter 2 strings u want to compare and check equal or not
abc
ab
Second string is smaller than first string
```



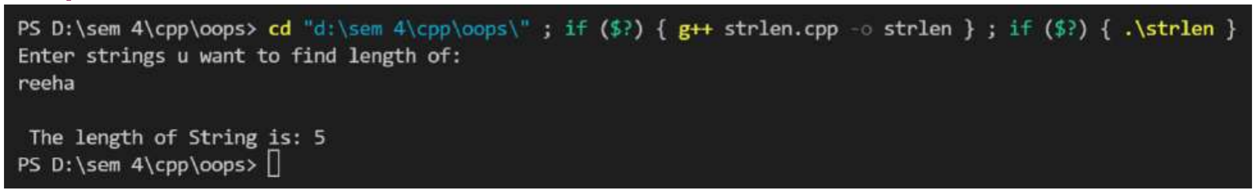
```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ comparisonStr.cpp -o comparisonStr } ; if ($?) { .\comparisonStr }
Enter 2 strings u want to compare and check equal or not
abc
ab
Second string is smaller than first string
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ comparisonStr.cpp -o comparisonStr } ; if ($?) { .\comparisonStr }
Enter 2 strings u want to compare and check equal or not
abc
abcd
First string is smaller than or equal to second
PS D:\sem 4\cpp\oops>
```

Function to display the length of string.

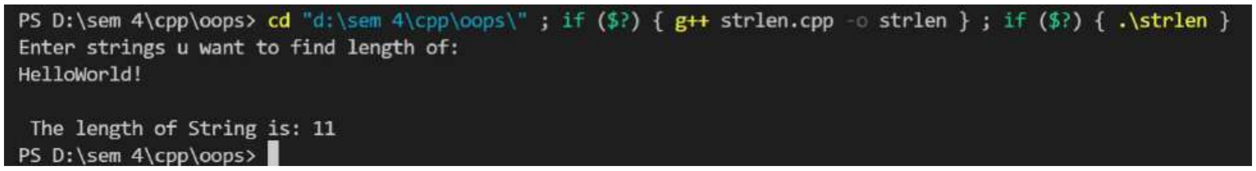
Source code

```
#include <iostream>
#include <cstring>

using namespace std;

class strLength
{
public:
    char str[25]; // Classes object of string

    void strlen()
    {
        cout << "Enter strings u want to find length of:" << endl;
        cin >> str;

        int l = 0;

        for (int i = 0; str[i] != '\0'; i++)
        {
            l++;
        }

        cout << "\n The length of String is: " << l;
    }
};

int main()
{
    strLength obj;
    obj.strlen();
    return 0;
}
```

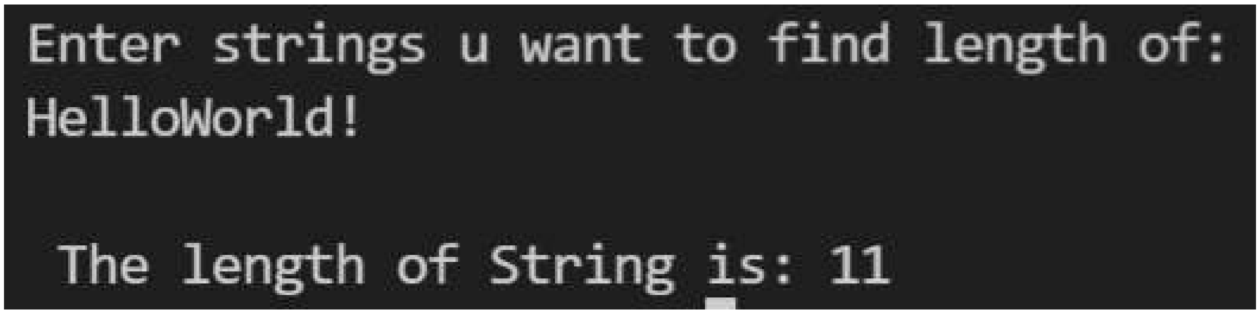
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strlen.cpp -o strlen } ; if ($?) { .\strlen }
Enter strings u want to find length of:
reeha

 The length of String is: 5
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ strlen.cpp -o strlen } ; if ($?) { .\strlen }
Enter strings u want to find length of:
HelloWorld!

 The length of String is: 11
PS D:\sem 4\cpp\oops> 
```



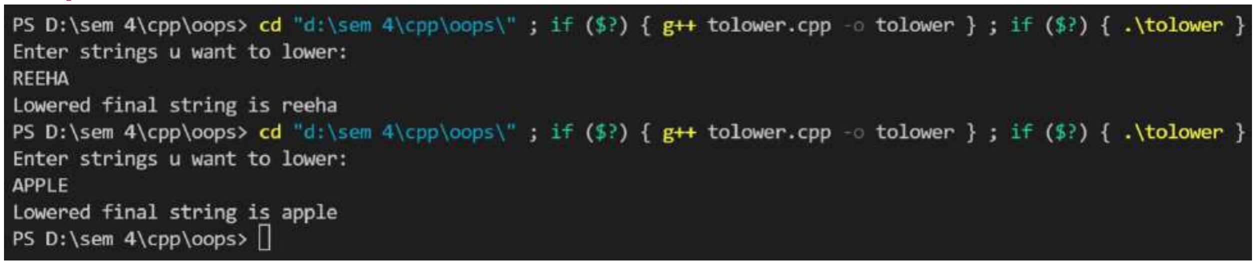
```
Enter strings u want to find length of:
HelloWorld!

 The length of String is: 11
```

Function tolower() to convert upper case to lower case.

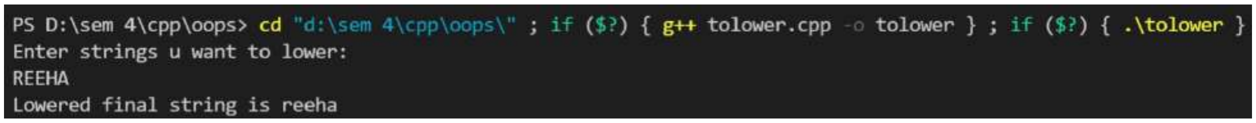
Source code

```
#include <iostream>
#include <cstring>

using namespace std;

class caseChange
{
public:
    char str[25]; // Classes object of string

    void tolower()
    {
        cout << "Enter strings u want to lower:" << endl;
        cin >> str;

        for (int i = 0; str[i] != '\0'; i++)
        {
            if ((str[i] >= 65) && (str[i] <= 90))
            {
                str[i] = str[i] + 32;
            }
        }
        cout << "Lowered final string is " << str;
    }
};

int main()
{
    caseChange obj;
    obj.tolower();
    return 0;
}
```

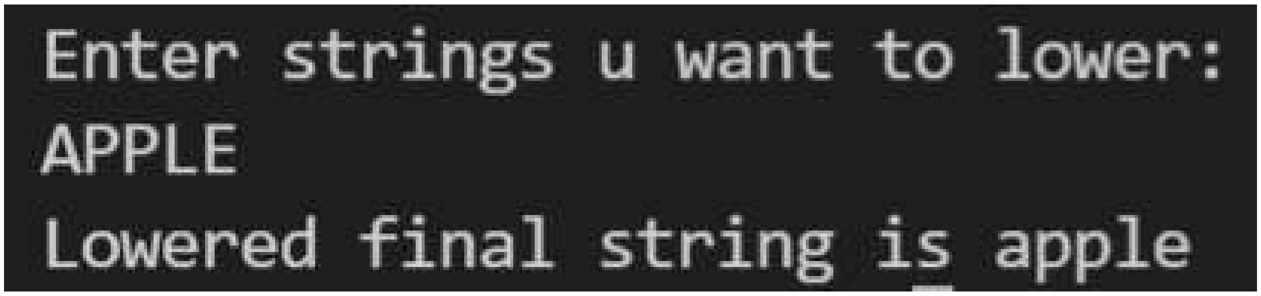
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tolower.cpp -o tolower } ; if ($?) { .\tolower }
Enter strings u want to lower:
REEHA
Lowered final string is reeha
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tolower.cpp -o tolower } ; if ($?) { .\tolower }
Enter strings u want to lower:
APPLE
Lowered final string is apple
PS D:\sem 4\cpp\oops> 
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tolower.cpp -o tolower } ; if ($?) { .\tolower }
Enter strings u want to lower:
REEHA
Lowered final string is reeha
```



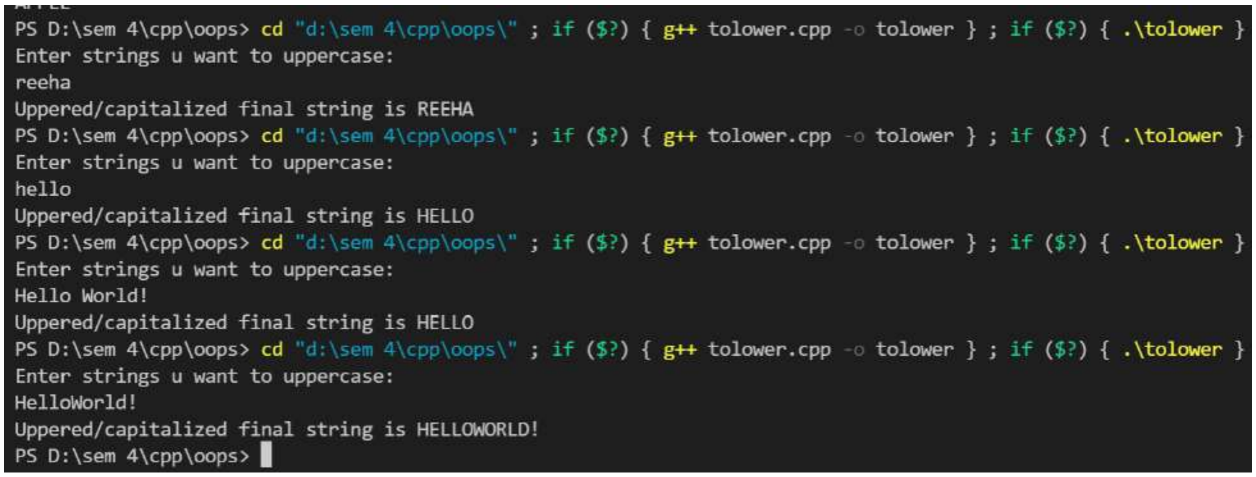
```
Enter strings u want to lower:
APPLE
Lowered final string is apple
```

Function toupper() to convert lower case letters to upper case.

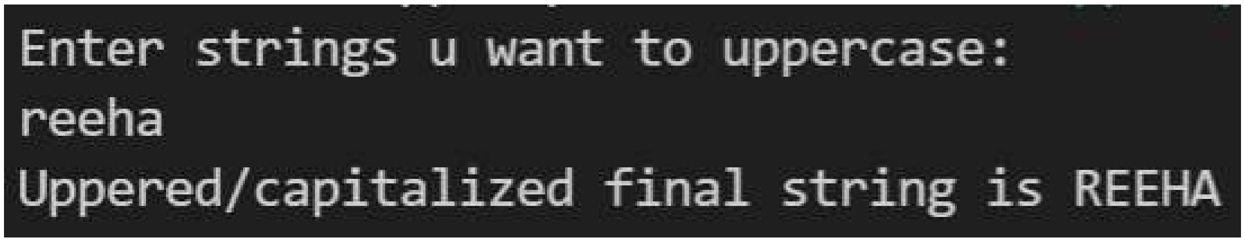
Source code

```
#include <iostream>
#include <cstring>

using namespace std;

class caseChange
{
public:
    char str[25]; // Classes object of string

    void toupper()
    {
        cout << "Enter strings u want to uppercase:" << endl;
        cin >> str;

        for (int i = 0; str[i] != '\0'; i++)
        {
            if ((str[i] >= 97) && (str[i] <= 122))
            {
                str[i] = str[i] - 32;
            }
        }

        cout << "Uppered/capitalized final string is " << str;
    }
};

int main()
{
    caseChange obj;
    obj.toupper();
    return 0;
}
```

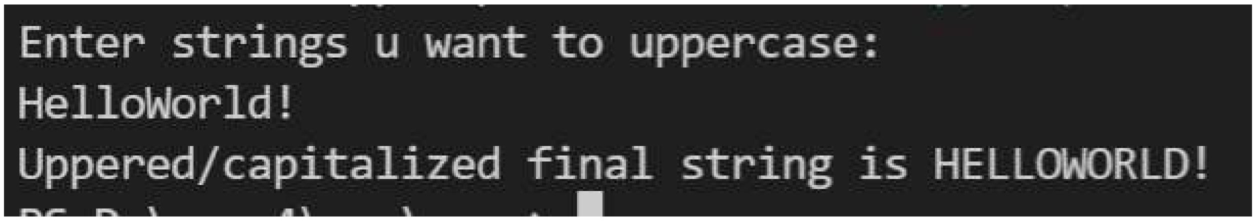
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tolower.cpp -o tolower } ; if ($?) { .\tolower }
Enter strings u want to uppercase:
reeha
Uppered/capitalized final string is REEHA
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tolower.cpp -o tolower } ; if ($?) { .\tolower }
Enter strings u want to uppercase:
hello
Uppered/capitalized final string is HELLO
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tolower.cpp -o tolower } ; if ($?) { .\tolower }
Enter strings u want to uppercase:
Hello World!
Uppered/capitalized final string is HELLO
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tolower.cpp -o tolower } ; if ($?) { .\tolower }
Enter strings u want to uppercase:
HelloWorld!
Uppered/capitalized final string is HELLOWORLD!
PS D:\sem 4\cpp\oops>
```



```
Enter strings u want to uppercase:
reeha
Uppered/capitalized final string is REEHA
```



```
Enter strings u want to uppercase:
HelloWorld!
Uppered/capitalized final string is HELLOWORLD!
```

Source code

```
#include <cstring>
#include <iostream>
#include <string.h>

using namespace std;

class operate
{
    char str[25]; // Classes object of string
public:
    void operator+(char s[25])
    {
        strcat(str, " ");
        strcat(str, s);
    }

    void operator=(char s[25])
    {
        strcpy(str, s);
    }

    void operator<=(char s[25])
    {
        if (strcmp(str, s) > 0)
        {
            cout << str << " is larger than " << s;
        }
        else if (strcmp(str, s) < 0)
        {
            cout << s << " is larger than " << str;
        }
        else
        {
            cout << str << " is equal to " << s;
        }
    }

    void strlen()
    {
        int l = 0;
        for (int i = 0; str[i] != '\0'; i++)
        {
            l++;
        }
        cout << "\n The length of String is: " << l;
    }
    void toupper()
    {
        for (int i = 0; str[i] != '\0'; i++)
        {
            if ((str[i] >= 97) && (str[i] <= 122))
            {
                str[i] = str[i] - 32;
            }
        }
    }
    void tolower()
    {
        for (int i = 0; str[i] != '\0'; i++)
        {
            if ((str[i] >= 65) && (str[i] <= 90))
            {
                str[i] = str[i] + 32;
            }
        }
    }

    void display()
    {
        cout << "Final string -----------> " << str << endl;
    }
};

int main()
{
    char s[100], st[100], ch;

    cout << "Enter 2 strings u want to compare and check equal or not" << endl;
    cin >> s >> st;

    operate r;
    cout << "\n\nChoose whether u want to \n1)string length \nconcat'+' \n2)to Up
per \ncompare'<' \n3) to lower \ncopy'=' \nexit(x)\n\n";
    //Invoking different operators
    cin >> ch;
    while (ch != 'x')
    {
        switch (ch)
        {
        case '+':
            r + st;
            r.display();
            break;

        case '=':
            r = st;
            r.display();
            break;

        case '<':
            r <= st;
            break;

        case 1:
            r.strlen();
            break;

        case 2:
            r.toupper();
            r.display();
            break;

        case 3:
            r.tolower();
            r.display();
            break;

        case 'x':
            cout << " Exit";
            exit(0);
        }
        cout << "\n\nChoose whether u want to \n1)string length \nconcat'+' \n2)t
o Upper \ncompare'<' \n3) to lower \ncopy'=' \nexit(x)\n\n";
        //Invoking different operators
        cin >> ch;
    }
    return 0;
}
```

```
// Operator overloading
            void strlen()
            {
                int l=0;
                for(int i=0;str[i]!='\0';i++)
                {
                 l++;
                }
                cout<<"\n The length of String is: "<<l;
            }
            void toupper()
            {
                for(int i=0;str[i]!='\0';i++)
                {
                 if((str[i]>=97)&&(str[i]<=122))
                 {
                    str[i]=str[i]-32;
                 }
                }
            }
            void tolower()            {
                for(int i=0;str[i]!='\0';i++)
                {
                 if((str[i]>=65)&&(str[i]<=90))
                 {
                    str[i]=str[i]+32;
                 }
                }
            }
```

```
void operator+(char s[25])
{
     strcat(str," ");
     strcat(str,s);
}
void operator=(char s[25])
{
     strcpy(str,s);
}
void operator<=(char s[25])
{
     if(strcmp(str,s)>0)
     {
       cout<<str<<" is larger than "<<s;
     }
     else if(strcmp(str,s)<0)
     {
       cout<<s<<" is larger than "<<str;
     }
     else
```

Viva Questions

Q1) What is overloading?

Ans.

*C++ allows you to specify more than one definition for a **function** name or an **operator** in the same scope, which is called **function overloading** and **operator overloading** respectively.*

An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).

*When you call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.*

Q2) What are causes of function overloading?

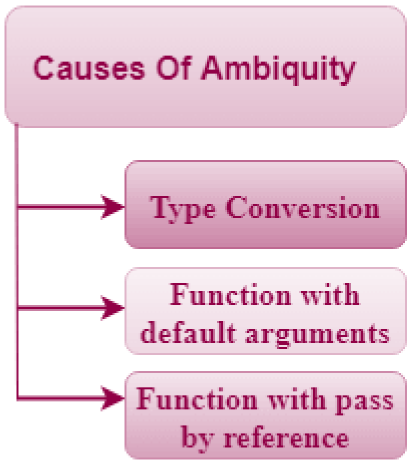
Ans.

When the compiler is unable to decide which function is to be invoked among the overloaded function, this situation is known as **function overloading**.

When the compiler shows the ambiguity error, the compiler does not run the program.

Causes of Function Overloading:

- Type Conversion.
- Function with default arguments.
- Function with pass by reference.


Causes Of Ambiquity
Type Conversion
Function with default arguments
Function with pass by reference

EXPERIMENT - 23

Object Oriented Programming Lab

Aim

Write a program to overload new and delete operator.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 23

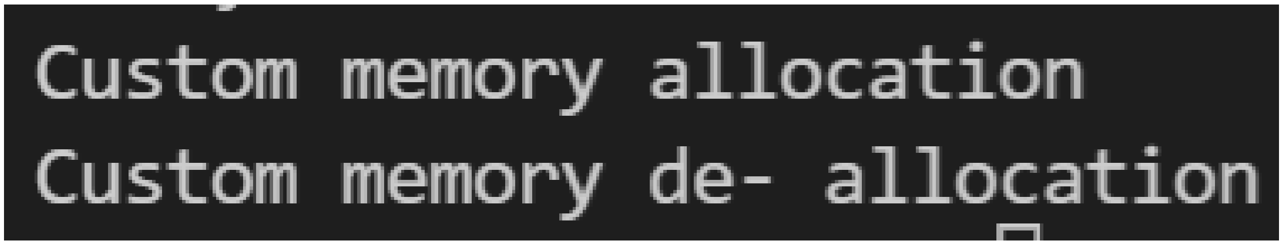
Aim:

Write a program to overload new and delete operator.

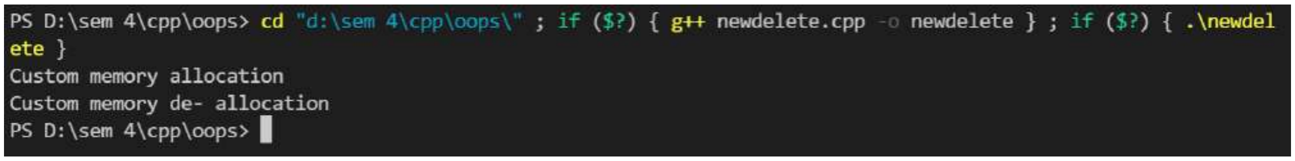
Source Code:

```
#include <iostream>
using namespace std;

class CustomMemory
{

public:
    void *operator new(size_t objectSize); //Overloaded new
    void operator delete(void *ptr);       //Overloaded delete
};

void *CustomMemory::operator new(size_t objectSize)
{
    cout << "Custom memory allocation" << endl;
    return malloc(objectSize);
}
void CustomMemory::operator delete(void *ptr)
{
    cout << "Custom memory de- allocation" << endl;
    free(ptr);
}

int main()
{

    // call overloaded new from the class
    CustomMemory *obj = new CustomMemory();

    // call overloaded delete
    delete obj;
}
```

Output:



```
Custom memory allocation
Custom memory de- allocation
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ newdelete.cpp -o newdelete } ; if ($?) { .\newdel
ete }
Custom memory allocation
Custom memory de- allocation
PS D:\sem 4\cpp\oops> 
```

Viva Questions

Q1) What are causes of function overloading?

Ans.

When the compiler is unable to decide which function is to be invoked among the overloaded function, this situation is known as **function overloading**.

When the compiler shows the ambiguity error, the compiler does not run the program.

Causes of Function Overloading:

- Type Conversion.
- Function with default arguments.
- Function with pass by reference.

Q2) What is overloading?

Ans.

C++ allows you to specify more than one definition for a **function** name or an **operator** in the same scope, which is called **function overloading** and **operator overloading** respectively.

An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).

*When you call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.*

Q3) What is purpose of operator overloading?

Ans.

The **purpose of operator overloading** is to provide a special meaning of an **operator** for a user-defined data type. With the help of **operator overloading**, you can redefine the majority of the C++ **operators**. You can also **use operator overloading** to perform different operations using one **operator**.

EXPERIMENT - 24

Object Oriented Programming Lab

Aim

Write a program to overload unary increment (++) operator.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 24

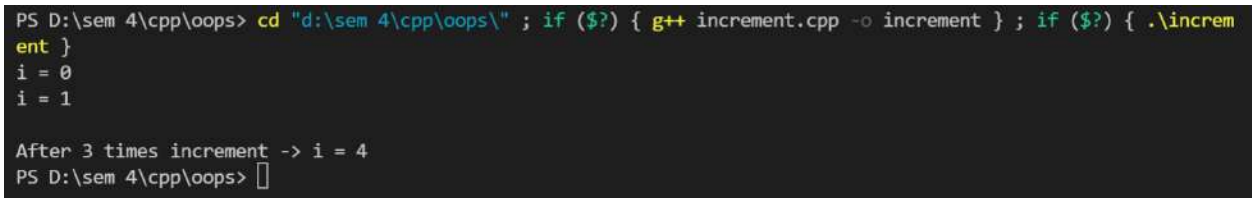
Aim:

Write a program to overload unary increment (++) operator.

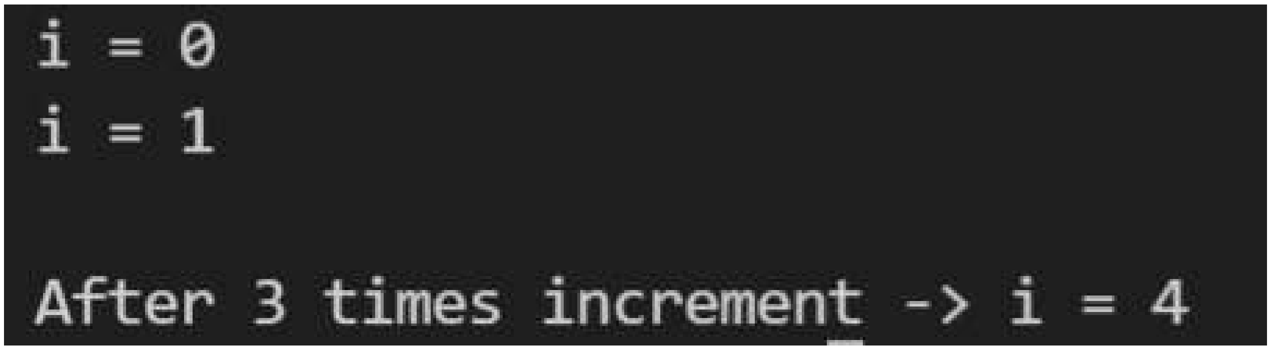
Source Code:

```
#include <iostream>
using namespace std;

class Check
{
private:
    int i;

public:
    Check() : i(0) {}
    void operator++()
    {
        ++i;
    }
    void Display()
    {
        cout << "i = " << i << endl;
    }
};

int main()
{
    Check obj;

    // Displays the value of data member i for object obj
    obj.Display();

    // Invokes operator function void operator ++( )
    ++obj;

    // Displays the value of data member i for object obj
    obj.Display();

    ++obj;
    ++obj;
    ++obj;
    cout << "\nAfter 3 times increment -> ";
    obj.Display();

    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ increment.cpp -o increment } ; if ($?) { .\increm
ent }
i = 0
i = 1

After 3 times increment -> i = 4
PS D:\sem 4\cpp\oops> 
```



```
i = 0
i = 1

After 3 times increment -> i = 4
```

Viva Questions

Q1) What are causes of function overloading?

Ans.

When the compiler is unable to decide which function is to be invoked among the overloaded function, this situation is known as **function overloading**.

When the compiler shows the ambiguity error, the compiler does not run the program.

Causes of Function Overloading:

- Type Conversion.
- Function with default arguments.
- Function with pass by reference.

Q2) What is overloading?

Ans.

C++ allows you to specify more than one definition for a **function** name or an **operator** in the same scope, which is called **function overloading** and **operator overloading** respectively.

An overloaded declaration is a declaration that is declared with the same name as a previously declared declaration in the same scope, except that both declarations have different arguments and obviously different definition (implementation).

*When you call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.*

Q3) What is purpose of operator overloading?

Ans.

The purpose of operator overloading is to provide a special meaning of an operator for a user-defined data type. With the help of operator overloading, you can redefine the majority of the C++ operators. You can also use operator overloading to perform different operations using one operator.

EXPERIMENT - 25

Object Oriented Programming Lab

Aim

Write a program to raise an exception if any attempt is made to refer to an element whose index is beyond the array size.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 25

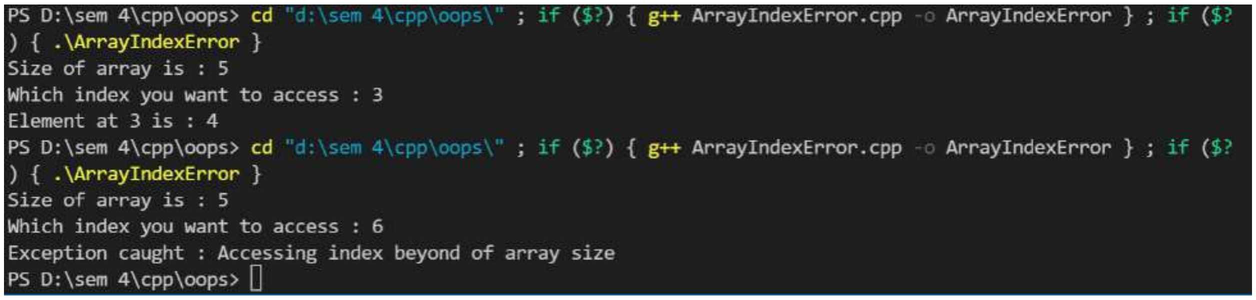
Aim:

Write a program to raise an exception if any attempt is made to refer to an element whose index is beyond the array size.

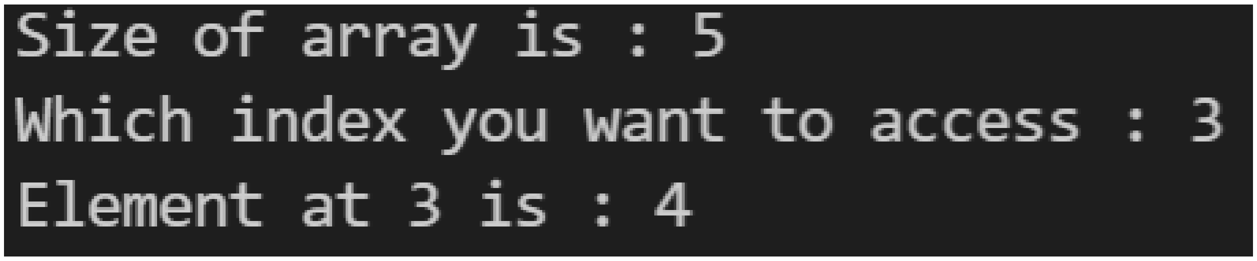
Source Code:

```
#include <iostream>
using namespace std;

int main()
{
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr) / sizeof(int);
    int i;
    cout << "Size of array is : " << n << endl;
    cout << "Which index you want to access : ";
    cin >> i;
    try
    {
        if (i >= n)
        {
            throw i;
        }
        else
        {
            cout << "Element at " << i << " is : " << arr[i];
        }
    }
    catch (int)
    {
        std::cout << "Exception caught : Accessing index beyond of array size";
    }

    return 0;
}
```

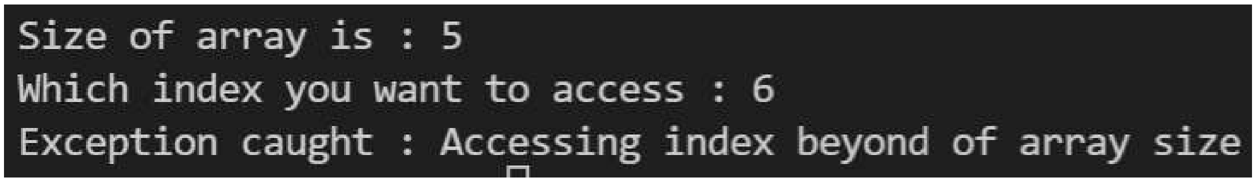
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ ArrayIndexError.cpp -o ArrayIndexError } ; if ($?
) { .\ArrayIndexError }
Size of array is : 5
Which index you want to access : 3
Element at 3 is : 4
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ ArrayIndexError.cpp -o ArrayIndexError } ; if ($?
) { .\ArrayIndexError }
Size of array is : 5
Which index you want to access : 6
Exception caught : Accessing index beyond of array size
PS D:\sem 4\cpp\oops>
```



```
Size of array is : 5
Which index you want to access : 3
Element at 3 is : 4
```



```
Size of array is : 5
Which index you want to access : 6
Exception caught : Accessing index beyond of array size
```

Viva Questions

Q1). What is an exception or error handling?

Ans.

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: **try, catch,** and **throw**.

- **throw** − A program throws an exception when a problem shows up. This is done using a **throw** keyword.
- **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The **catch** keyword indicates the catching of an exception.
- **try** − A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the **try** and **catch** keywords. A try/catch block is placed around the code that might generate an exception.

Q2). What does throw exception mean?

Ans.

Exceptions can be thrown anywhere within a code block using **throw** statement. The operand of the throw statement determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.

Q3). What does catch exception mean?

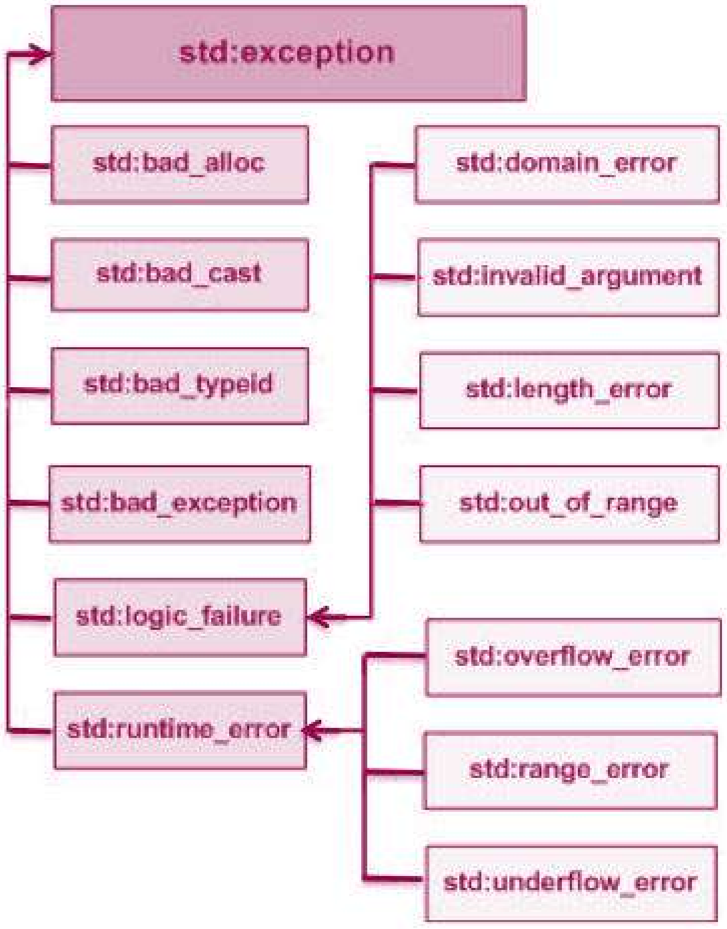
Ans.

The **catch** block following the **try** block catches any exception. You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch.

Q4). What are some standard exceptions in cpp?

Ans.

C++ provides a list of standard exceptions defined in **<exception>** which we can use in our programs. These are arranged in a parent-child class hierarchy shown below –



EXPERIMENT - 26

Object Oriented Programming Lab

Aim

Write a program to read two numbers and then divide first no by second no and raise an exception if second number is zero.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 26

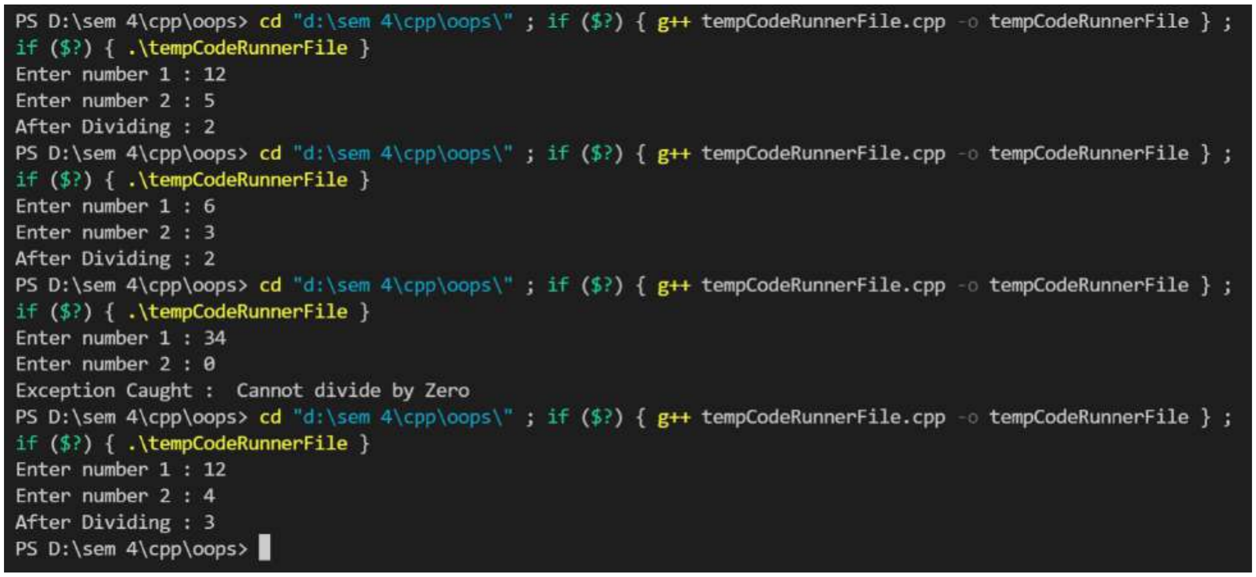
Aim:

Write a program to read two numbers and then divide first no by second no and raise an exception if second number is zero.

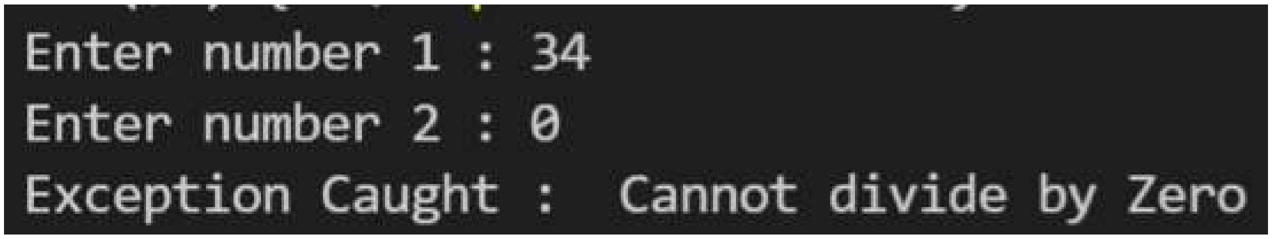
Source Code:

```
#include <iostream>

using namespace std;


int main()
{
    int x, y;
    cout << "Enter number 1 : ";
    cin >> x;
    cout << "Enter number 2 : ";
    cin >> y;
    try
    {
        if (y == 0)
        {
            throw y;
        }
        else
        {
            cout << "After Dividing : " << (x / y);
        }
    }
    catch (int)
    {
        cout << "Exception Caught :  Cannot divide by Zero";
    }
    return 0;
}
```

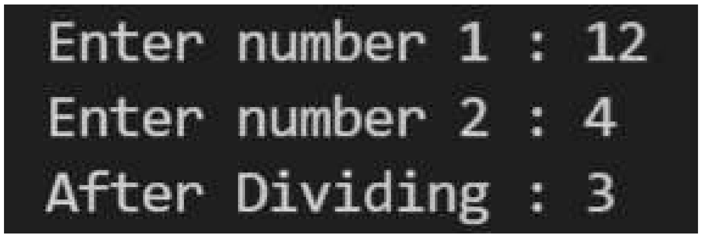
Output:



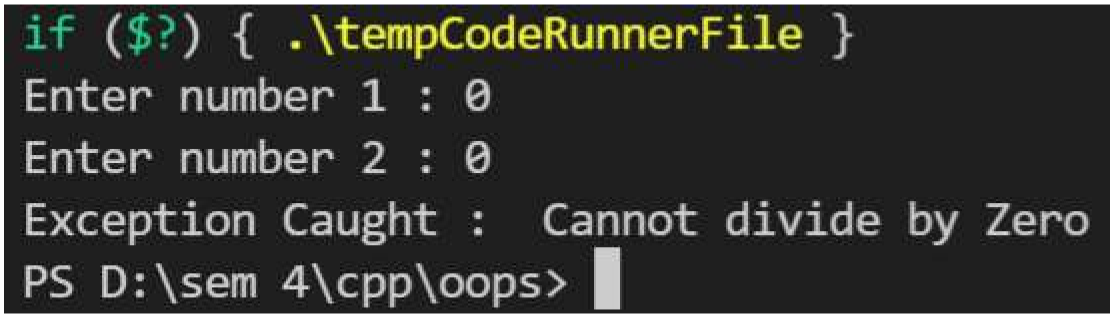
```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ;
if ($?) { .\tempCodeRunnerFile }
Enter number 1 : 12
Enter number 2 : 5
After Dividing : 2
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ;
if ($?) { .\tempCodeRunnerFile }
Enter number 1 : 6
Enter number 2 : 3
After Dividing : 2
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ;
if ($?) { .\tempCodeRunnerFile }
Enter number 1 : 34
Enter number 2 : 0
Exception Caught :  Cannot divide by Zero
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ;
if ($?) { .\tempCodeRunnerFile }
Enter number 1 : 12
Enter number 2 : 4
After Dividing : 3
PS D:\sem 4\cpp\oops>
```



```
Enter number 1 : 34
Enter number 2 : 0
Exception Caught :  Cannot divide by Zero
```



```
Enter number 1 : 12
Enter number 2 : 4
After Dividing : 3
```



```
if ($?) { .\tempCodeRunnerFile }
Enter number 1 : 0
Enter number 2 : 0
Exception Caught :  Cannot divide by Zero
PS D:\sem 4\cpp\oops>
```

Viva Questions

Q1). What is an exception or error handling?

Ans.

An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: **try, catch,** and **throw**.

- **throw** − A program throws an exception when a problem shows up. This is done using a **throw** keyword.
- **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The **catch** keyword indicates the catching of an exception.
- **try** − A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the **try** and **catch** keywords. A try/catch block is placed around the code that might generate an exception.

Q2). What does throw exception mean?

Ans.

Exceptions can be thrown anywhere within a code block using **throw** statement. The operand of the throw statement determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.

Q3). What does catch exception mean?

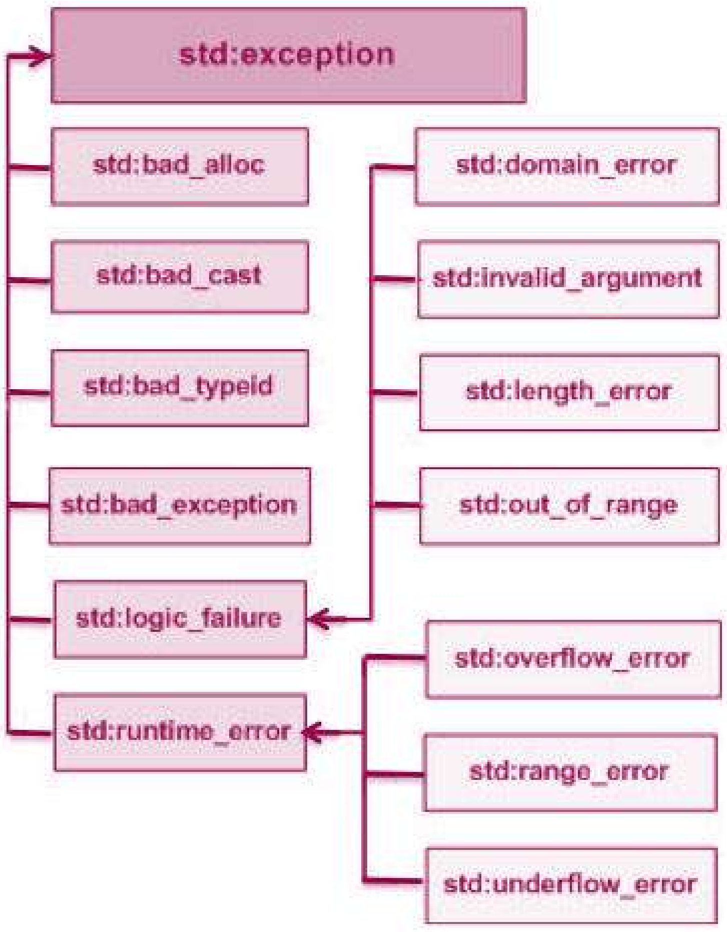
Ans.

The **catch** block following the **try** block catches any exception. You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch.

Q4). What are some standard exceptions in cpp?

Ans.

C++ provides a list of standard exceptions defined in **<exception>** which we can use in our programs. These are arranged in a parent-child class hierarchy shown below –



EXPERIMENT - 27

Object Oriented Programming Lab

Aim

Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space, line feed, new line and carriage return from a text file and store the contents of the file without the white spaces on another file.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 27

Aim:

Write a program to perform the deletion of white spaces such as horizontal tab, vertical tab, space, line feed, new line and carriage return from a text file and store the contents of the file without the white spaces on another file.

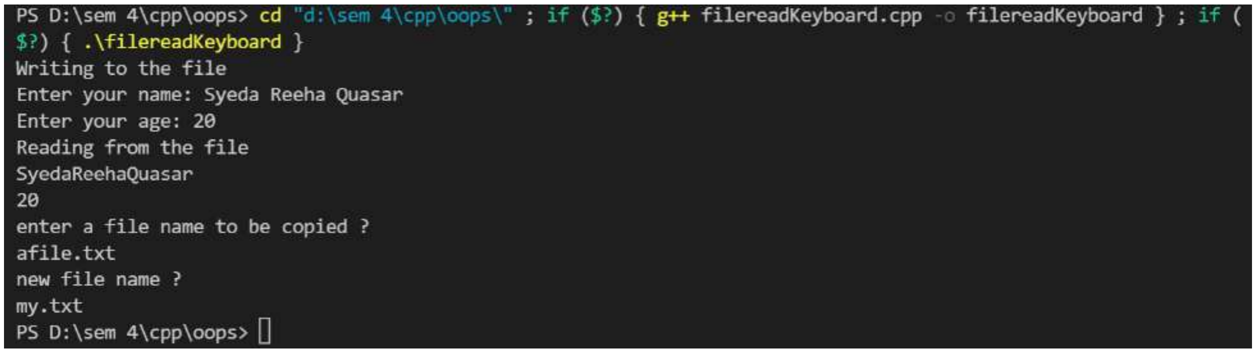
Source Code:

```
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    char data[100];

    // open a file in write mode.
    ofstream outfile;
    outfile.open("afile.txt");

    cout << "Writing to the file" << endl;
    cout << "Enter your name: ";
    cin.getline(data, 100);

    // write inputted data into the file.
    outfile << data << endl;

    cout << "Enter your age: ";
    cin >> data;
    cin.ignore();

    // again write inputted data into the file.
    outfile << data << endl;

    // close the opened file.
    outfile.close();
```

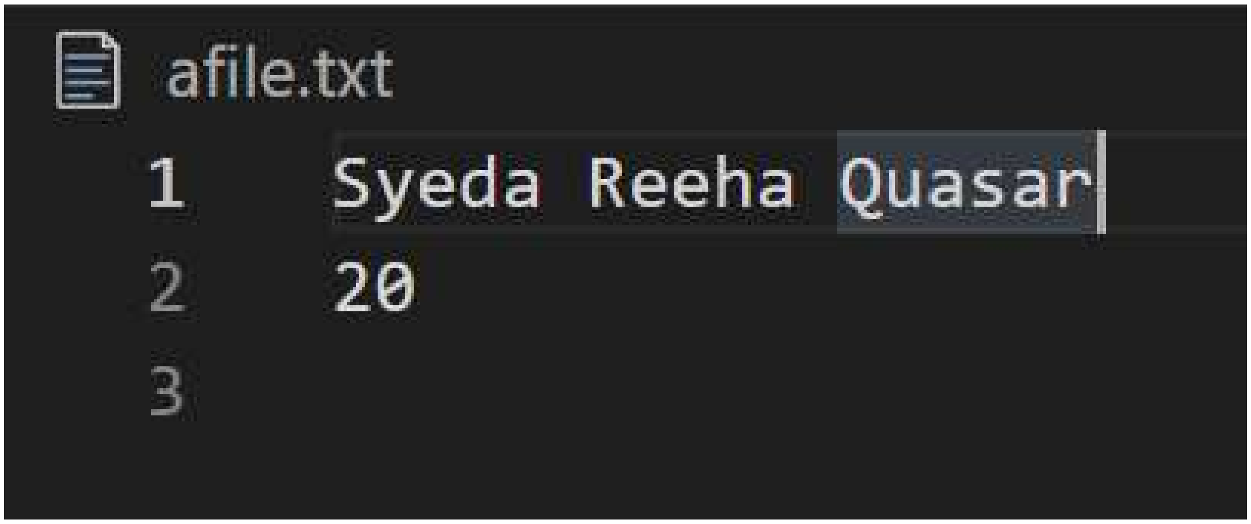
```
// open a file in read mode.
ifstream infile;
infile.open("afile.txt");

cout << "Reading from the file" << endl;
infile >> data;

// write the data at the screen.
cout << data;

infile >> data;
cout << data;

infile >> data;
cout << data << endl;

// again read the data from the file and display it.
infile >> data;
cout << data << endl;

// close the opened file.
infile.close();

char fname1[10], fname2[10];
char ch;

cout << "enter a file name to be copied ?\n";
cin >> fname1;

cout << "new file name ? \n";
cin >> fname2;

infile.open(fname1);

if (infile.fail())
{
    cerr << "No such a file exists \n";
    exit(1);
}
outfile.open(fname2);

if (outfile.fail())
{
    cerr <<"unable to create a file \n";
```

```
        exit(1);
    }

    while (!infile.eof())
    {
        ch = (char)infile.get();
        if (ch == ' ' || ch == '\t' || ch == '\n');
        else
            outfile.put(ch);
    }

    // close the opened file.
    infile.close();

    // close the opened file.
    outfile.close();

    return 0;
}
```

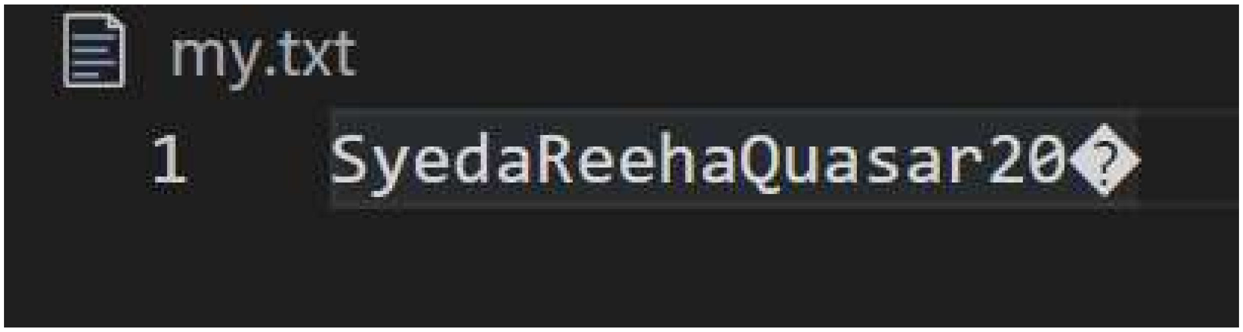
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ filereadKeyboard.cpp -o filereadKeyboard } ; if (
$?) { .\filereadKeyboard }
Writing to the file
Enter your name: Syeda Reeha Quasar
Enter your age: 20
Reading from the file
SyedaReehaQuasar
20
enter a file name to be copied ?
afile.txt
new file name ?
my.txt
PS D:\sem 4\cpp\oops> 
```



```
afile.txt
1   Syeda Reeha Quasar
2   20
3
```



```
my.txt
1   SyedaReehaQuasar20�
```

Viva Questions

Q1). What is file handling in C++?

Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file.

A stream is an abstraction of a device where input/output operations are performed. You can represent a stream as either a destination or a source of characters of indefinite length.

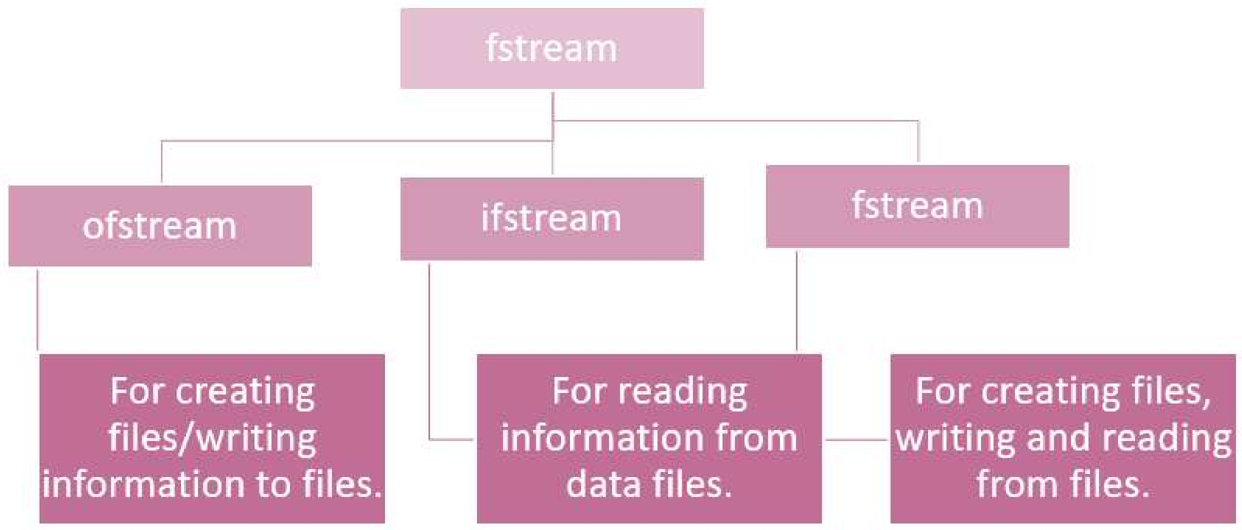
Q2). What is fstream Library?

Ans.

The fstream library provides C++ programmers with three classes for working with files. These classes include:

- **ofstream**- This class represents an output stream. It's used for creating files and writing information to files.
- **ifstream**- This class represents an input stream. It's used for reading information from data files.
- **fstream**- This class generally represents a file stream. It comes with ofstream/ifstream capabilities. This means it's capable of creating files, writing to files, reading from data files.

The following image makes it simple to understand:


fstream
ofstream
ifstream
fstream
For creating files/writing information to files.
For reading information from data files.
For creating files, writing and reading from files.

fstream library

To use the above classes of the fstream library, you must include it in your program as a header file. Of course, you will use the #include preprocessor directive. You must also include the iostream header file.

Q3). What are benefits of file handing?

Ans.

- With file handling, the output of a program can be sent and stored in a file.
- A number of operations can then be applied to the data while in the file.
- A stream is an abstraction that represents a device where input/output operations are performed.
- A stream can be represented as either destination or source of characters of indefinite length.
- The fstream library provides C++ programmers with methods for file handling.
- To use the library, you must include it in your program using the #include preprocessor directive.

EXPERIMENT - 28

Object Oriented Programming Lab

Aim

Write a program to define the function template for calculating the square of given numbers with different data types.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 28

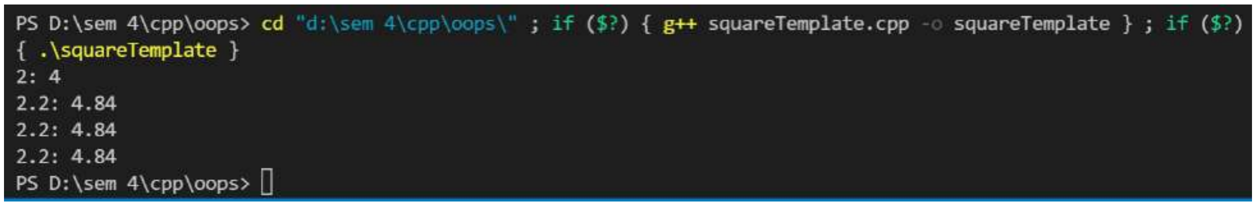
Aim:

Write a program to define the function template for calculating the square of given numbers with different data types.

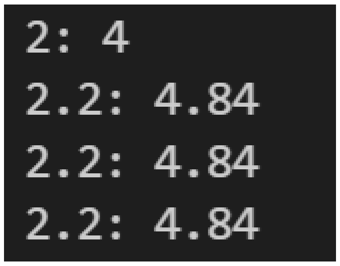
Source Code:

```
#include <iostream>


using namespace std;


template <class T>
inline T square(T x)
{
    T result;
    result = x * x;
    return result;
};


int main()
{
    int i, ii;
    float x, xx;
    double y, yy;

    i = 2;
    x = 2.2;
    y = 2.2;
    ii = square<int>(i);
    cout << i << ": " << ii << endl;

    xx = square<float>(x);
    cout << x << ": " << xx << endl;

    // Explicit use of template
    yy = square<double>(y);
    cout << y << ": " << yy << endl;

    // Implicit use of template
    yy = square(y);
    cout << y << ": " << yy << endl;
    return 0;
}
```

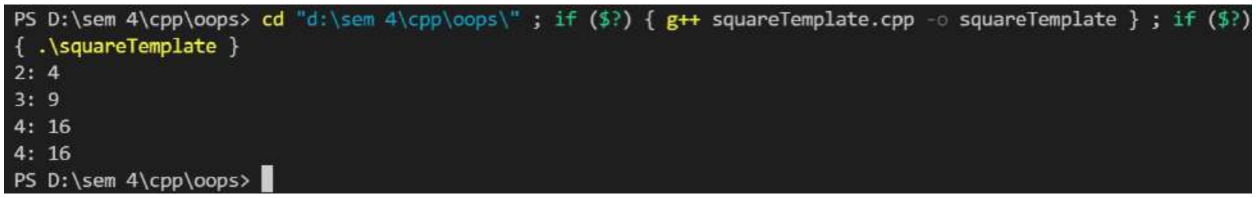
Output:



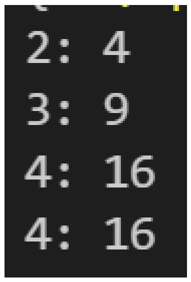
```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ squareTemplate.cpp -o squareTemplate } ; if ($?) { .\squareTemplate }
2: 4
2.2: 4.84
2.2: 4.84
2.2: 4.84
PS D:\sem 4\cpp\oops>
```



```
2: 4
2.2: 4.84
2.2: 4.84
2.2: 4.84
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ squareTemplate.cpp -o squareTemplate } ; if ($?) { .\squareTemplate }
2: 4
3: 9
4: 16
4: 16
PS D:\sem 4\cpp\oops>
```



```
2: 4
3: 9
4: 16
4: 16
```

Viva Questions

Q1). What are templates in C++?

Ans.

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

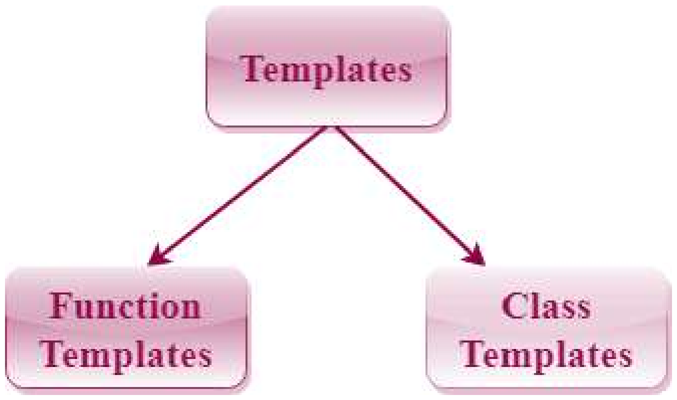
There is a single definition of each container, such as **vector**, but we can define many different kinds of vectors for example, **vector <int>** or **vector <string>**.

Q2). How can templates be classified?

Ans.

Templates can be represented in two ways:

- Function templates
- Class templates



Function Templates:

We can define a template for a function. For example, if we have an add() function, we can create versions of the add function for adding the int, float or double type values.

Class Template:

We can define a template for a class. For example, a class template can be created for the array class that can accept the array of various types such as int array, float array or double array.

Q3). Write about Function templates.

Ans.

- C++ supports a powerful feature known as a template to implement the concept of generic programming.
- A template allows us to create a family of classes or family of functions to handle different data types.
- Template classes and functions eliminate the code duplication of different data types and thus makes the development easier and faster.
- Multiple parameters can be used in both class and function template.
- Template functions can also be overloaded.
- We can also use nontype arguments such as built-in or derived data types as template arguments.

EXPERIMENT - 29

Object Oriented Programming Lab

Aim

Write a program to define the function template for swapping two items of various datatypes such as integers, float and characters.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 29

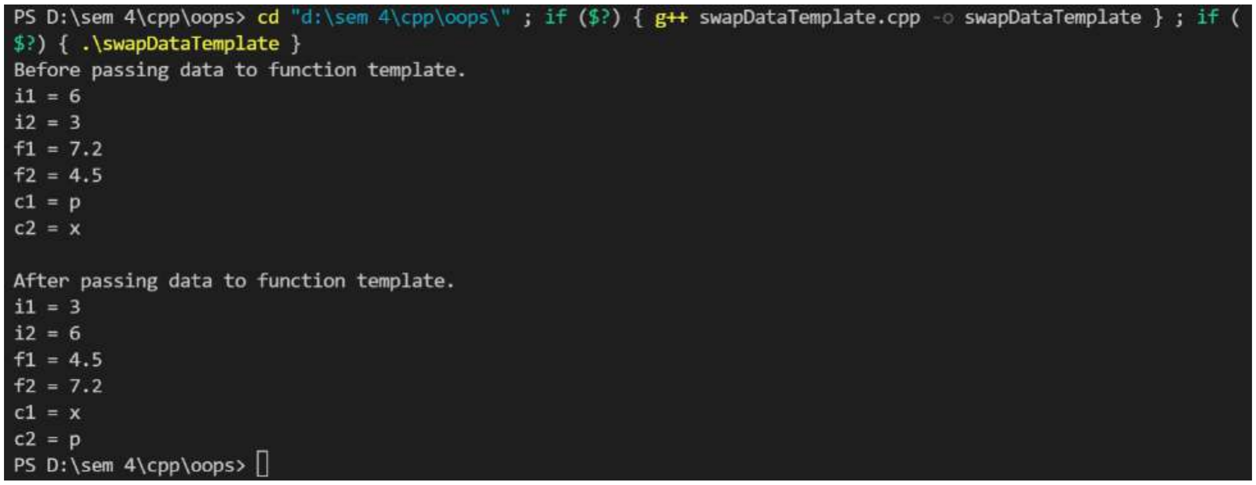
Aim:

Write a program to define the function template for swapping two items of various datatypes such as integers, float and characters.

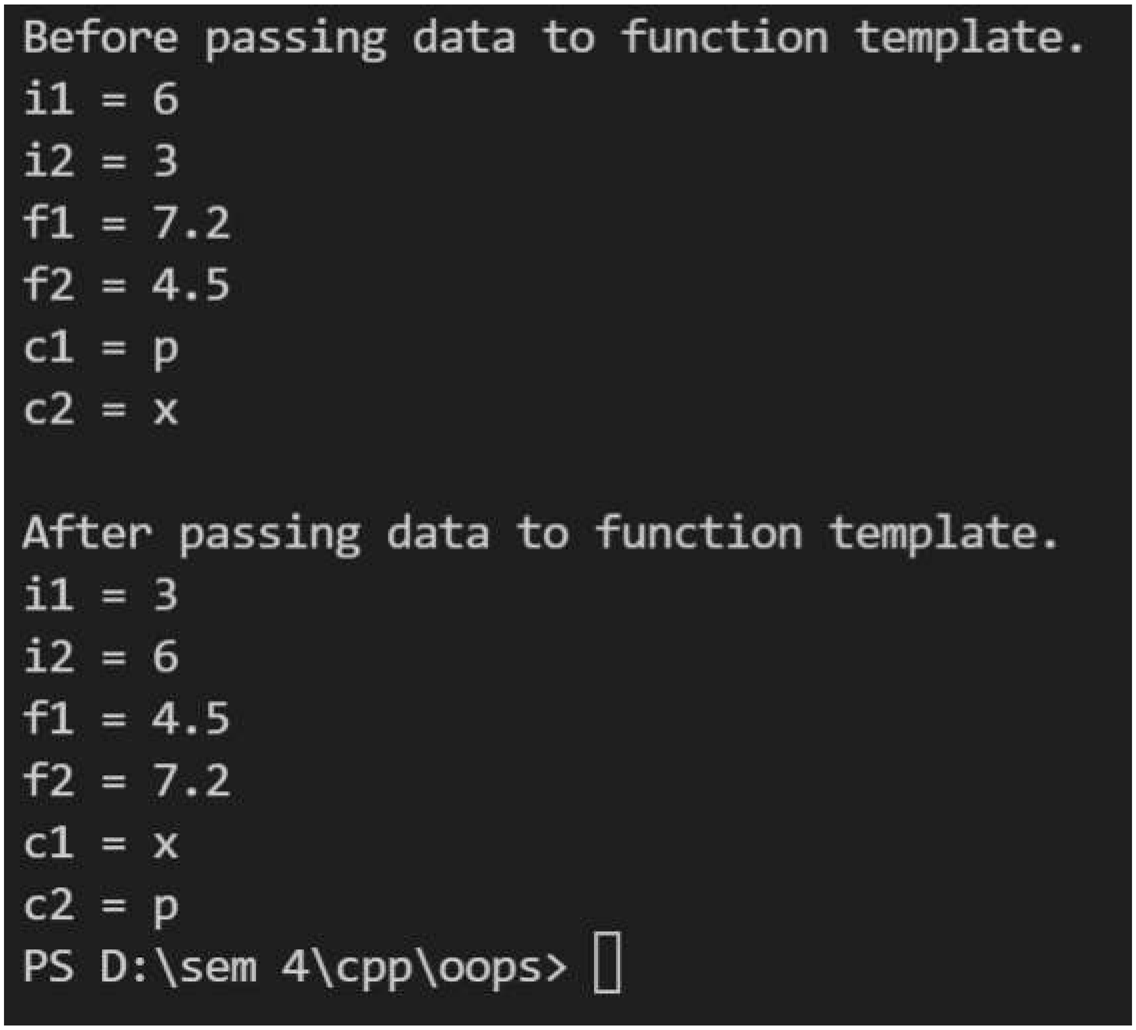
Source Code:

```
#include <iostream>
using namespace std;

template <typename T>
void Swap(T &n1, T &n2)
{
    T temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
}

int main()
{
    int i1 = 6, i2 = 3;
    float f1 = 7.2, f2 = 4.5;
    char c1 = 'p', c2 = 'x';

    cout << "Before passing data to function template.\n";
    cout << "i1 = " << i1 << "\ni2 = " << i2;
    cout << "\nf1 = " << f1 << "\nf2 = " << f2;
    cout << "\nc1 = " << c1 << "\nc2 = " << c2;

    Swap(i1, i2);
    Swap(f1, f2);
    Swap(c1, c2);

    cout << "\n\nAfter passing data to function template.\n";
    cout << "i1 = " << i1 << "\ni2 = " << i2;
    cout << "\nf1 = " << f1 << "\nf2 = " << f2;
    cout << "\nc1 = " << c1 << "\nc2 = " << c2;

    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ swapDataTemplate.cpp -o swapDataTemplate } ; if (
$?) { .\swapDataTemplate }
Before passing data to function template.
i1 = 6
i2 = 3
f1 = 7.2
f2 = 4.5
c1 = p
c2 = x

After passing data to function template.
i1 = 3
i2 = 6
f1 = 4.5
f2 = 7.2
c1 = x
c2 = p
PS D:\sem 4\cpp\oops> 
```



```
Before passing data to function template.
i1 = 6
i2 = 3
f1 = 7.2
f2 = 4.5
c1 = p
c2 = x

After passing data to function template.
i1 = 3
i2 = 6
f1 = 4.5
f2 = 7.2
c1 = x
c2 = p
PS D:\sem 4\cpp\oops> 
```

Viva Questions

Q1). What are templates in C++?

Ans.

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

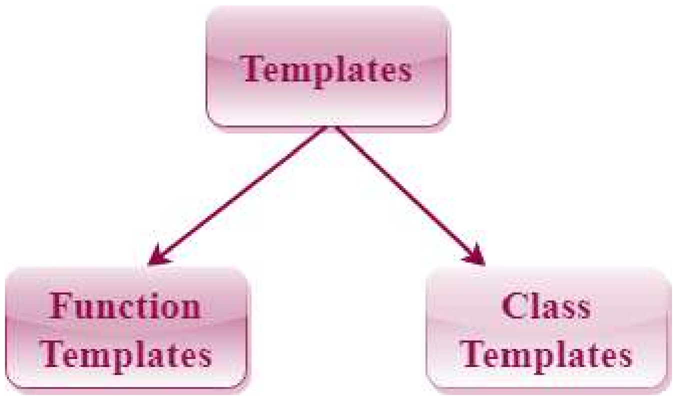
There is a single definition of each container, such as **vector**, but we can define many different kinds of vectors for example, **vector <int>** or **vector <string>**.

Q2). How can templates be classified?

Ans.

Templates can be represented in two ways:

- Function templates
- Class templates



Function Templates:

We can define a template for a function. For example, if we have an add() function, we can create versions of the add function for adding the int, float or double type values.

Class Template:

We can define a template for a class. For example, a class template can be created for the array class that can accept the array of various types such as int array, float array or double array.

Q3). Write about Function templates.

Ans.

- C++ supports a powerful feature known as a template to implement the concept of generic programming.
- A template allows us to create a family of classes or family of functions to handle different data types.
- Template classes and functions eliminate the code duplication of different data types and thus makes the development easier and faster.
- Multiple parameters can be used in both class and function template.
- Template functions can also be overloaded.
- We can also use nontype arguments such as built-in or derived data types as template arguments.

Q4). What are different Data types?

Ans.

Data Type	Meaning	Size (in Bytes)
`int`	Integer	2 or 4
`float`	Floating-point	4
`double`	Double Floating-point	8
`char`	Character	1
`wchar_t`	Wide Character	2
`bool`	Boolean	1
`void`	Empty	0

EXPERIMENT - 30

Object Oriented Programming Lab

Aim

Write a program to illustrate how Template function can be overloaded.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 30

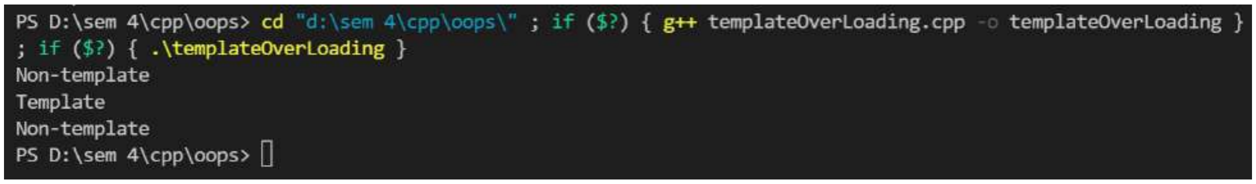
Aim:

Write a program to illustrate how Template function can be overloaded.

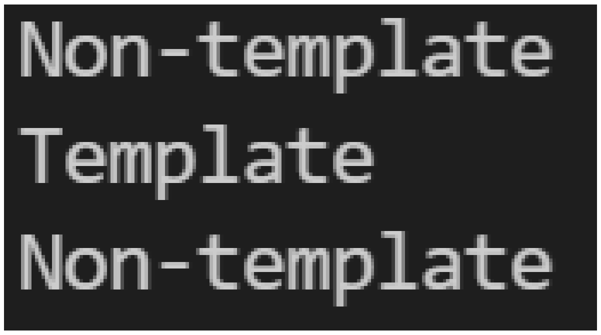
Source Code:

```
#include <iostream>

using namespace std;

template <class T>
void f(T x, T y) { cout << "Template" << endl; }

void f(int w, int z) { cout << "Non-template" << endl; }

int main()
{
    f(1, 2);
    f('a', 'b');
    f(1, 'b');
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ templateOverLoading.cpp -o templateOverLoading }
; if ($?) { .\templateOverLoading }
Non-template
Template
Non-template
PS D:\sem 4\cpp\oops> 
```



```
Non-template
Template
Non-template
```

Viva Questions

Q1). What are templates in C++?

Ans.

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

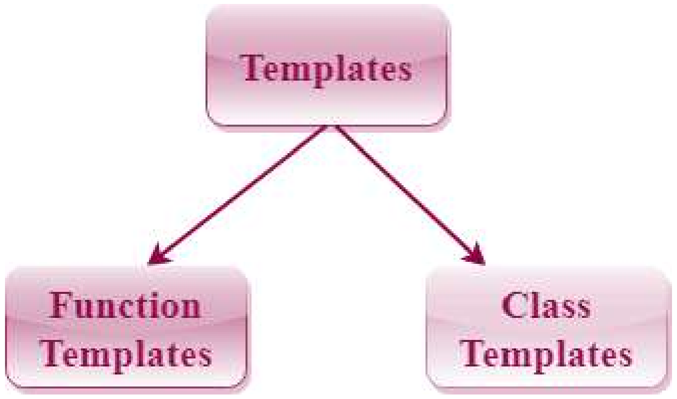
There is a single definition of each container, such as **vector**, but we can define many different kinds of vectors for example, **vector <int>** or **vector <string>**.

Q2). How can templates be classified?

Ans.

Templates can be represented in two ways:

- Function templates
- Class templates



Function Templates:

We can define a template for a function. For example, if we have an add() function, we can create versions of the add function for adding the int, float or double type values.

Class Template:

We can define a template for a class. For example, a class template can be created for the array class that can accept the array of various types such as int array, float array or double array.

Q3). Write about Function templates.

Ans.

- C++ supports a powerful feature known as a template to implement the concept of generic programming.
- A template allows us to create a family of classes or family of functions to handle different data types.
- Template classes and functions eliminate the code duplication of different data types and thus makes the development easier and faster.
- Multiple parameters can be used in both class and function template.
- Template functions can also be overloaded.
- We can also use nontype arguments such as built-in or derived data types as template arguments.

Q3) What is purpose of operator overloading?

Ans.

The purpose of operator overloading is to provide a special meaning of an operator for a user-defined data type. With the help of operator overloading, you can redefine the majority of the C++ operators. You can also use operator overloading to perform different operations using one operator.

EXPERIMENT - 31

Object Oriented Programming Lab

Aim

Write a program to illustrate how to define and declare a class template for reading two data items from the keyboard and to find their sum.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 31

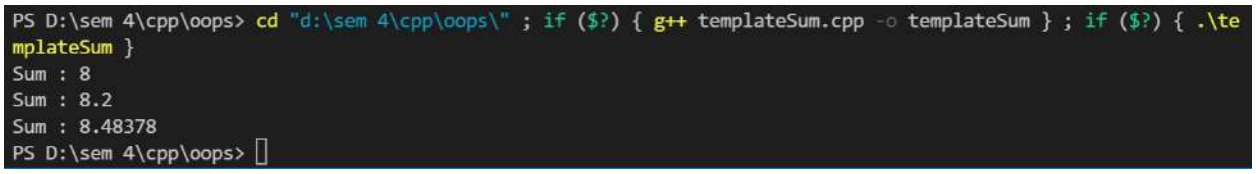
Aim:

Write a program to illustrate how to define and declare a class template for reading two data items from the keyboard and to find their sum.

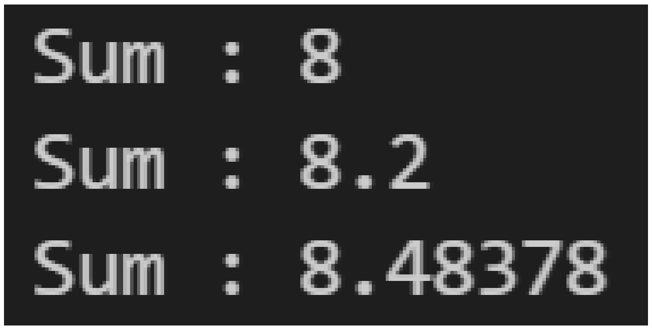
Source Code:

```
#include <iostream>


using namespace std;


template <typename T>
T sum(T x, T y)
{
   return x + y;
}


int main()
{
   cout << "Sum : " << sum(3, 5) << endl;
   cout << "Sum : " << sum(3.0, 5.2) << endl;
   cout << "Sum : " << sum(3.24234, 5.24144) << endl;
   return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ templateSum.cpp -o templateSum } ; if ($?) { .\te
mplateSum }
Sum : 8
Sum : 8.2
Sum : 8.48378
PS D:\sem 4\cpp\oops> 
```



```
Sum : 8
Sum : 8.2
Sum : 8.48378
```

Viva Questions

Q1). What are templates in C++?

Ans.

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type.

A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept.

There is a single definition of each container, such as **vector**, but we can define many different kinds of vectors for example, **vector <int>** or **vector <string>**.

Q2). How can templates be classified?

Ans.

Function Templates:

We can define a template for a function. For example, if we have an add() function, we can create versions of the add function for adding the int, float or double type values.

Class Template:

We can define a template for a class. For example, a class template can be created for the array class that can accept the array of various types such as int array, float array or double array.

Q3). Write about Function templates.

Ans.

- C++ supports a powerful feature known as a template to implement the concept of generic programming.
- A template allows us to create a family of classes or family of functions to handle different data types.

- Template classes and functions eliminate the code duplication of different data types and thus makes the development easier and faster.
- Multiple parameters can be used in both class and function template.
- Template functions can also be overloaded.
- We can also use nontype arguments such as built-in or derived data types as template arguments.

Q4) What is purpose of operator overloading?

Ans.

The purpose of operator overloading is to provide a special meaning of an operator for a user-defined data type. With the help of operator overloading, you can redefine the majority of the C++ operators. You can also use operator overloading to perform different operations using one operator.

Q5). What are the benefits of OOPs?

Ans.

The procedural-oriented languages focus on procedures, with function as the basic unit. You need to first figure out all the functions and then think about how to represent data.

The object-oriented languages focus on components that the user perceives, with objects as the basic unit. You figure out all the objects by putting all the data and operations that describe the user's interaction with the data.

Object-Oriented technology has many benefits:

- *Ease in software design* as you could think in the problem space rather than the machine's bits and bytes. You are dealing with high-level concepts and abstractions. Ease in design leads to more productive software development.
- *Ease in software maintenance*: object-oriented software are easier to understand, therefore easier to test, debug, and maintain.
- *Reusable software*: you don't need to keep re-inventing the wheels and re-write the same functions for different situations. The fastest and safest way of developing a new application is to reuse existing codes - fully tested and proven codes.

EXPERIMENT - 32

Object Oriented Programming Lab

Aim

Write a program to read a set of lines from the keyboard and to store it on a specified file.

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 32

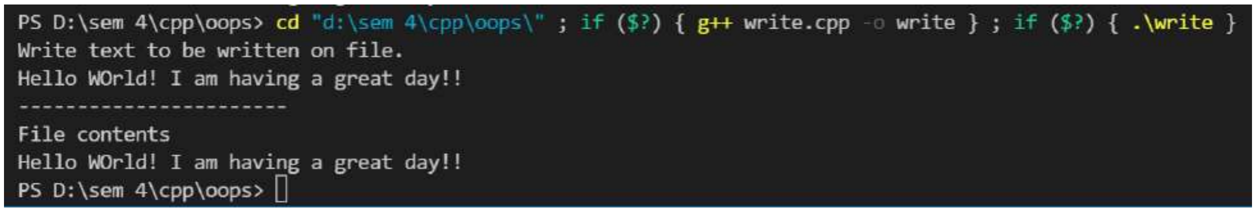
Aim:

Write a program to read a set of lines from the keyboard and to store it on a specified file.

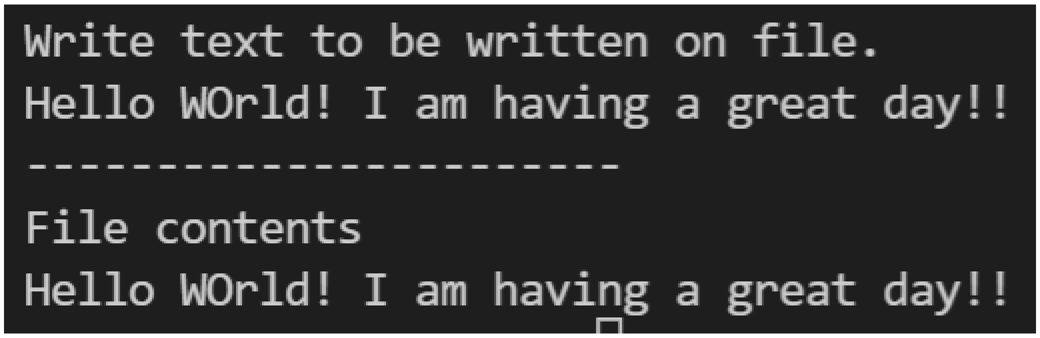
Source Code:

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{

    char text[200];

    fstream file;
    file.open("1.txt", ios::out | ios::in);

    cout << "Write text to be written on file." << endl;
    cin.getline(text, sizeof(text));

    // Writing on file
    file << text << endl;

    // Reding from file
    cout << "-----------------------\nFile contents\n";
    file >> text;
    cout << text << endl;

    //closing the file
    file.close();
    return 0;
}
```

Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ write.cpp -o write } ; if ($?) { .\write }
Write text to be written on file.
Hello WOrld! I am having a great day!!
-----------------------
File contents
Hello WOrld! I am having a great day!!
PS D:\sem 4\cpp\oops> 
```



```
Write text to be written on file.
Hello WOrld! I am having a great day!!
-----------------------
File contents
Hello WOrld! I am having a great day!!
```

Viva Questions

Q1). What is file handling in C++?

Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file.

A stream is an abstraction of a device where input/output operations are performed. You can represent a stream as either a destination or a source of characters of indefinite length.

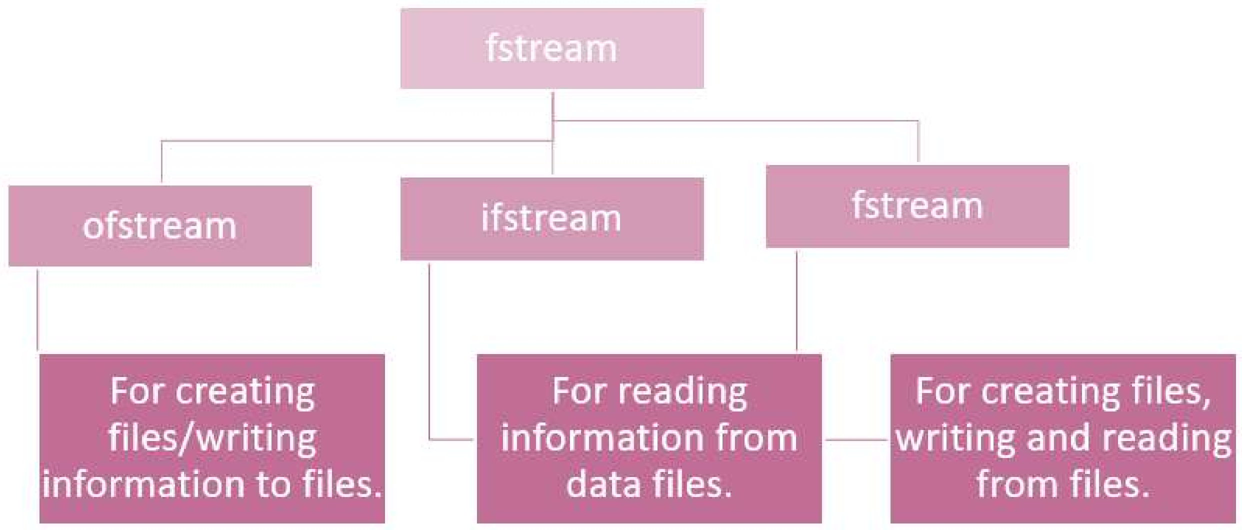
Q2). What is fstream Library?

Ans.

The fstream library provides C++ programmers with three classes for working with files. These classes include:

- **ofstream**- This class represents an output stream. It's used for creating files and writing information to files.
- **ifstream**- This class represents an input stream. It's used for reading information from data files.
- **fstream**- This class generally represents a file stream. It comes with ofstream/ifstream capabilities. This means it's capable of creating files, writing to files, reading from data files.

The following image makes it simple to understand:


fstream
ofstream
ifstream
fstream
For creating files/writing information to files.
For reading information from data files.
For creating files, writing and reading from files.

fstream library

To use the above classes of the fstream library, you must include it in your program as a header file. Of course, you will use the #include preprocessor directive. You must also include the iostream header file.

Q3). What are benefits of file handing?

Ans.

- With file handling, the output of a program can be sent and stored in a file.
- A number of operations can then be applied to the data while in the file.
- A stream is an abstraction that represents a device where input/output operations are performed.
- A stream can be represented as either destination or source of characters of indefinite length.
- The fstream library provides C++ programmers with methods for file handling.
- To use the library, you must include it in your program using the #include preprocessor directive.

Q4). What are the benefits of OOPs?

Ans.

The procedural-oriented languages focus on procedures, with function as the basic unit. You need to first figure out all the functions and then think about how to represent data.

The object-oriented languages focus on components that the user perceives, with objects as the basic unit. You figure out all the objects by putting all the data and operations that describe the user's interaction with the data.

Object-Oriented technology has many benefits:

- *Ease in software design* as you could think in the problem space rather than the machine's bits and bytes. You are dealing with high-level concepts and abstractions. Ease in design leads to more productive software development.
- *Ease in software maintenance*: object-oriented software are easier to understand, therefore easier to test, debug, and maintain.
- *Reusable software*: you don't need to keep re-inventing the wheels and re-write the same functions for different situations. The fastest and safest way of developing a new application is to reuse existing codes - fully tested and proven codes.

EXPERIMENT - 33

Object Oriented Programming Lab

Aim

Write a program to read a text file and display its contents on the screen.

Syeda Reeha Quasar
14114802719
4C7

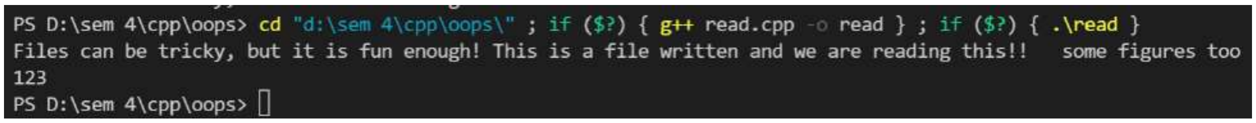
EXPERIMENT – 33

Aim:
Write a program to read a text file and display its contents on the screen.

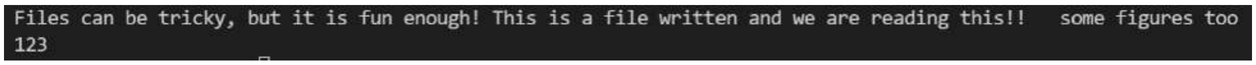
Source Code:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main () {
  // Create a text file
  ofstream MyWriteFile("filename.txt");

  // Write to the file
  MyWriteFile << "Files can be tricky, but it is fun enough! This is a file writt
en and we are reading this!! \n \n some figures too 123 \n\n";

  // Close the file
  MyWriteFile.close();

  // Create a text string, which is used to output the text file
  string myText;

  // Read from the text file
  ifstream MyReadFile("filename.txt");

  // Use a while loop together with the getline() function to read the file line
by line
  while (getline (MyReadFile, myText)) {
    // Output the text from the file
    cout << myText;
  }

  // Close the file
  MyReadFile.close();
}
```

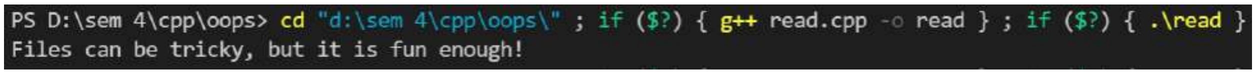
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ read.cpp -o read } ; if ($?) { .\read }
Files can be tricky, but it is fun enough! This is a file written and we are reading this!!   some figures too
123
PS D:\sem 4\cpp\oops> 
```



```
Files can be tricky, but it is fun enough! This is a file written and we are reading this!!   some figures too
123
```



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ read.cpp -o read } ; if ($?) { .\read }
Files can be tricky, but it is fun enough!
```

Viva Questions

Q1). What is file handling in C++?

Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file.

A stream is an abstraction of a device where input/output operations are performed. You can represent a stream as either a destination or a source of characters of indefinite length.

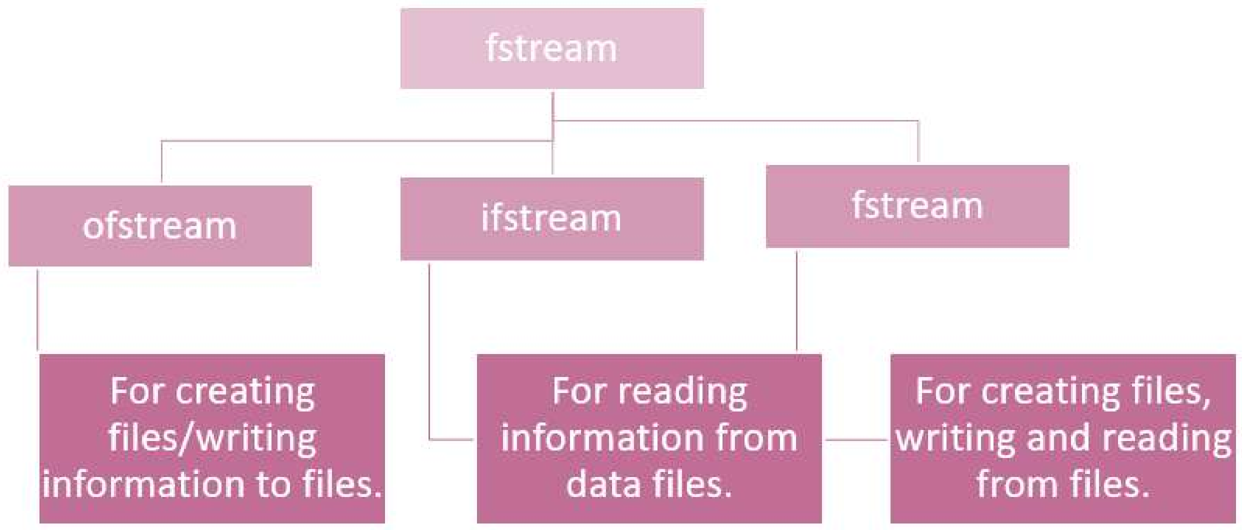
Q2). What is fstream Library?

Ans.

The fstream library provides C++ programmers with three classes for working with files. These classes include:

- **ofstream**- This class represents an output stream. It's used for creating files and writing information to files.
- **ifstream**- This class represents an input stream. It's used for reading information from data files.
- **fstream**- This class generally represents a file stream. It comes with ofstream/ifstream capabilities. This means it's capable of creating files, writing to files, reading from data files.

The following image makes it simple to understand:


fstream
ofstream
ifstream
fstream
For creating files/writing information to files.
For reading information from data files.
For creating files, writing and reading from files.

fstream library

To use the above classes of the fstream library, you must include it in your program as a header file. Of course, you will use the #include preprocessor directive. You must also include the iostream header file.

Q3). What are benefits of file handing?

Ans.

- With file handling, the output of a program can be sent and stored in a file.
- A number of operations can then be applied to the data while in the file.
- A stream is an abstraction that represents a device where input/output operations are performed.
- A stream can be represented as either destination or source of characters of indefinite length.
- The fstream library provides C++ programmers with methods for file handling.
- To use the library, you must include it in your program using the #include preprocessor directive.

Q4). What are the benefits of OOPs?

Ans.

The procedural-oriented languages focus on procedures, with function as the basic unit. You need to first figure out all the functions and then think about how to represent data.

The object-oriented languages focus on components that the user perceives, with objects as the basic unit. You figure out all the objects by putting all the data and operations that describe the user's interaction with the data.

Object-Oriented technology has many benefits:

- *Ease in software design* as you could think in the problem space rather than the machine's bits and bytes. You are dealing with high-level concepts and abstractions. Ease in design leads to more productive software development.
- *Ease in software maintenance*: object-oriented software are easier to understand, therefore easier to test, debug, and maintain.
- *Reusable software*: you don't need to keep re-inventing the wheels and re-write the same functions for different situations. The fastest and safest way of developing a new application is to reuse existing codes - fully tested and proven codes.

EXPERIMENT - 34

Object Oriented Programming Lab

Aim

Write a program to copy the contents of a file into another..

Syeda Reeha Quasar
14114802719
4C7

EXPERIMENT – 34

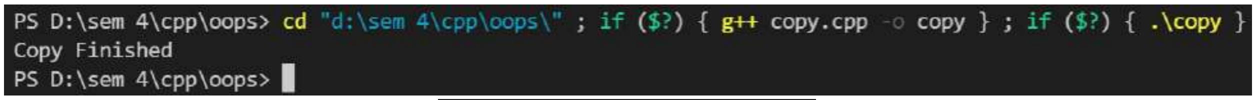
Aim:

Write a program to copy the contents of a file into another.

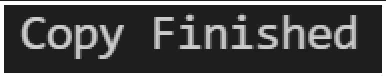
Source Code:

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    string line;
    //For writing text file
    //Creating ofstream & ifstream class object
    ifstream ini_file{"filename.txt"};
    ofstream out_file{"copy.txt"};

    if (ini_file && out_file)
    {

        while (getline(ini_file, line))
        {
            out_file << line << "\n";
        }

        cout << "Copy Finished \n";
    }
    else
    {
        //Something went wrong
        printf("Cannot read File");
    }

    //Closing file
    ini_file.close();
    out_file.close();
    return 0;
}
```

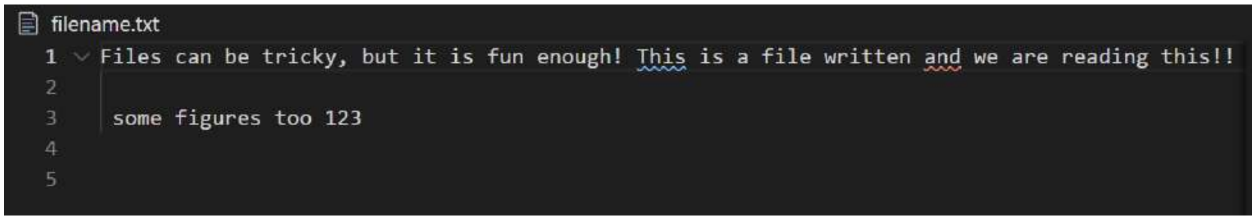
Output:



```
PS D:\sem 4\cpp\oops> cd "d:\sem 4\cpp\oops\" ; if ($?) { g++ copy.cpp -o copy } ; if ($?) { .\copy }
Copy Finished
PS D:\sem 4\cpp\oops> 
```

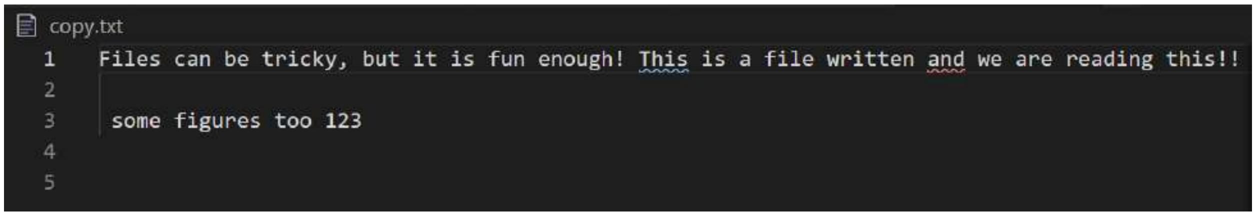


```
Copy Finished
```



filename.txt

```
Files can be tricky, but it is fun enough! This is a file written and we are reading this!!

 some figures too 123


```



copy.txt

```
Files can be tricky, but it is fun enough! This is a file written and we are reading this!!

 some figures too 123


```

Viva Questions

Q1). What is file handling in C++?

Files store data permanently in a storage device. With file handling, the output from a program can be stored in a file. Various operations can be performed on the data while in the file.

A stream is an abstraction of a device where input/output operations are performed. You can represent a stream as either a destination or a source of characters of indefinite length.

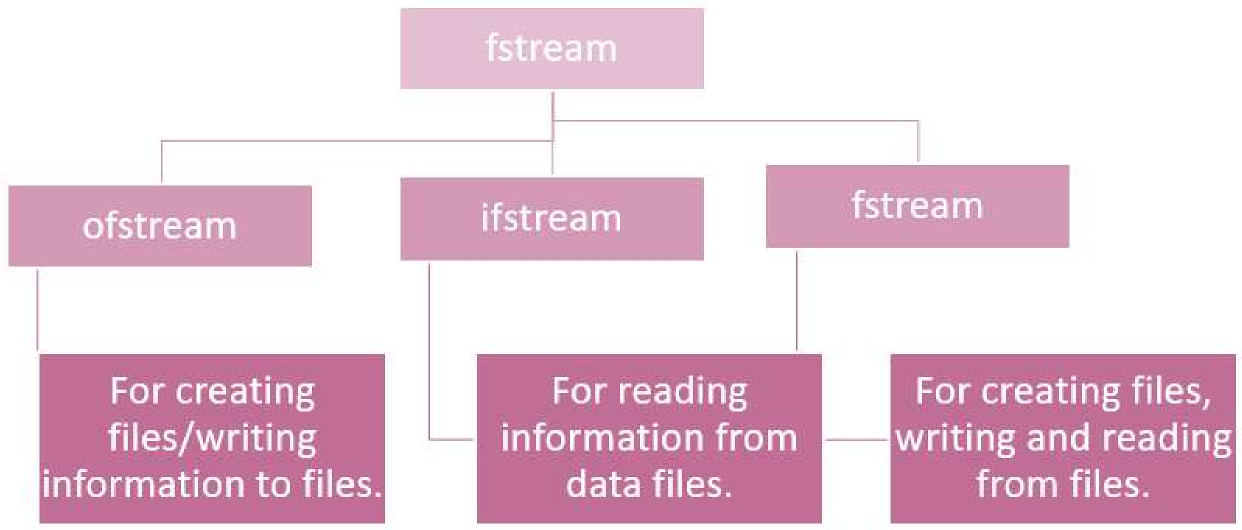
Q2). What is fstream Library?

Ans.

The fstream library provides C++ programmers with three classes for working with files. These classes include:

- **ofstream**- This class represents an output stream. It's used for creating files and writing information to files.
- **ifstream**- This class represents an input stream. It's used for reading information from data files.
- **fstream**- This class generally represents a file stream. It comes with ofstream/ifstream capabilities. This means it's capable of creating files, writing to files, reading from data files.

The following image makes it simple to understand:


fstream
ofstream
ifstream
fstream
For creating files/writing information to files.
For reading information from data files.
For creating files, writing and reading from files.

fstream library

To use the above classes of the fstream library, you must include it in your program as a header file. Of course, you will use the #include preprocessor directive. You must also include the iostream header file.

Q3). What are benefits of file handing?

Ans.

- With file handling, the output of a program can be sent and stored in a file.
- A number of operations can then be applied to the data while in the file.
- A stream is an abstraction that represents a device where input/output operations are performed.
- A stream can be represented as either destination or source of characters of indefinite length.
- The fstream library provides C++ programmers with methods for file handling.
- To use the library, you must include it in your program using the #include preprocessor directive.

Q4). What are the benefits of OOPs?

Ans.

The procedural-oriented languages focus on procedures, with function as the basic unit. You need to first figure out all the functions and then think about how to represent data.

The object-oriented languages focus on components that the user perceives, with objects as the basic unit. You figure out all the objects by putting all the data and operations that describe the user's interaction with the data.

Object-Oriented technology has many benefits:

- *Ease in software design* as you could think in the problem space rather than the machine's bits and bytes. You are dealing with high-level concepts and abstractions. Ease in design leads to more productive software development.
- *Ease in software maintenance*: object-oriented software are easier to understand, therefore easier to test, debug, and maintain.
- *Reusable software*: you don't need to keep re-inventing the wheels and re-write the same functions for different situations. The fastest and safest way of developing a new application is to reuse existing codes - fully tested and proven codes.